BS IEC 61523-4:2023

BSI Standards Publication

Delay and power calculation standards

Part 4: Design and Verification of Low-Power, Energy-Aware Electronic Systems

Documents
Document Sharing Hub

Full document details: BS IEC 61523-4:2023

National foreword

This British Standard is the UK implementation of IEC 61523-4:2023. It supersedes BS IEC 61523-4:2015, which is withdrawn.

The UK participation in its preparation was entrusted to Technical Committee EPL/501, Electronic Assembly Technology.

A list of organizations represented on this committee can be obtained on request to its committee manager.

Contractual and legal considerations

This publication has been prepared in good faith, however no representation, warranty, assurance or undertaking (express or implied) is or will be made, and no responsibility or liability is or will be accepted by BSI in relation to the adequacy, accuracy, completeness or reasonableness of this publication. All and any such responsibility and liability is expressly disclaimed to the full extent permitted by the law.

This publication is provided as is, and is to be used at the recipient's own risk.

The recipient is advised to consider seeking professional guidance with respect to its use of this publication.

This publication is not intended to constitute a contract. Users are responsible for its correct application.

© The British Standards Institution 2024
Published by BSI Standards Limited 2024

ISBN 978 0 539 27839 2

ICS 25.040.01; 35.060

Compliance with a British Standard cannot confer immunity from legal obligations.

This British Standard was published under the authority of the Standards Policy and Strategy Committee on 29 February 2024.

Amendments/corrigenda issued since publication

Date	Text affected

Documents
Document Sharing Hub

Full document details: BS IEC 61523-4:2023

IEC 61523-4

Edition 2.0 2023-09

INTERNATIONAL STANDARD

IEEE Std 1801™

Delay and power calculation standards – Part 4: Design and Verification of Low-Power, Energy-Aware Electronic Systems

INTERNATIONAL
ELECTROTECHNICAL
COMMISSION

ICS 25.040.01, 35.060

ISBN 978-2-8322-7540-5

Warning! Make sure that you obtained this publication from an authorized distributor.

® Registered trademark of the International Electrotechnical Commission

Documents
Document Sharing Hub

Full document details: BS IEC 61523-4:2023

Contents

1. Overview 13
1.1 Scope 13
1.2 Purpose 13
1.3 Key characteristics of the Unified Power Format 13
1.4 Contents of this standard 15

2. Normative references 16

3. Definitions, acronyms, and abbreviations 16
3.1 Definitions 16
3.2 Acronyms and abbreviations 22

4. Concepts 23
4.1 Introduction 23
4.2 Design structure 24
4.3 Design representation 24
4.4 Power architecture 28
4.5 Power distribution 31
4.6 Power management 39
4.7 Supply states and power states 44
4.8 Simstates 51
4.9 Power intent specification 52

5. Language basics 58
5.1 UPF is Tcl 58
5.2 Conventions used 59
5.3 Lexical elements 61
5.4 Boolean expressions 65
5.5 Object declaration 67
5.6 Attributes of objects 67
5.7 Precedence 72
5.8 Generic UPF command semantics 75
5.9 effective_element_list semantics 76
5.10 Command refinement 79
5.11 Error handling 80
5.12 Units 80
5.13 SystemC language basic 80

6. Power intent commands 81
6.1 Introduction 81
6.2 Categories 81
6.3 add_parameter 82
6.4 add_port_state (legacy) 83
6.5 add_power_state 84
6.6 add_pst_state (legacy) 91
6.7 add_state_transition 92
6.8 add_supply_state 94
6.9 apply_power_model 95
6.10 associate_supply_set 97
6.11 begin_power_model (legacy) 99
6.12 bind_checker 100

Documents
Document Sharing Hub
Published by IEC under licence from IEEE. © 2018 IEEE. All rights reserved.
Full document details: BS IEC 61523-4:2023

6.13 connect_logic_net 102
6.14 connect_supply_net 104
6.15 connect_supply_set 106
6.16 create_composite_domain 107
6.17 create_hdl2upf_vct 109
6.18 create_logic_net 110
6.19 create_logic_port 111
6.20 create_power_domain 112
6.21 create_power_state_group 119
6.22 create_power_switch 121
6.23 create_pst (legacy) 128
6.24 create_supply_net 129
6.25 create_supply_port 133
6.26 create_supply_set 134
6.27 create_upf2hdl_vct 136
6.28 define_power_model 137
6.29 describe_state_transition (deprecated) 139
6.30 end_power_model (legacy) 139
6.31 find_objects 140
6.32 load_simstate_behavior 144
6.33 load_upf 145
6.34 load_upf_protected (deprecated) 146
6.35 map_power_switch 146
6.36 map_repeater_cell 147
6.37 map_retention_cell 148
6.38 name_format 152
6.39 save_upf 153
6.40 set_correlated 154
6.41 set_design_attributes 155
6.42 set_design_top 156
6.43 set_domain_supply_net (legacy) 157
6.44 set_equivalent 158
6.45 set_isolation 160
6.46 set_level_shifter 167
6.47 set_partial_on_translation 173
6.48 set_port_attributes 175
6.49 set_repeater 181
6.50 set_retention 185
6.51 set_retention_elements 189
6.52 set_scope 190
6.53 set_simstate_behavior 191
6.54 set_variation 194
6.55 sim_assertion_control 195
6.56 sim_corruption_control 197
6.57 sim_replay_control 200
6.58 upf_version 202
6.59 use_interface_cell 203

7. Power-management cell definition commands 205
7.1 Introduction 205
7.2 define_always_on_cell 206
7.3 define_diode_clamp 207
7.4 define_isolation_cell 208
7.5 define_level_shifter_cell 211
7.6 define_power_switch_cell 216
7.7 define_retention_cell 218

Published by IEC under licence from IEEE. © 2018 IEEE. All rights reserved.

Documents
Document Sharing Hub

Full document details: BS IEC 61523-4:2023

8. UPF processing 220
8.1 Overview 220
8.2 Data requirements 221
8.3 Processing phases 221
8.4 Error checking 225

9. Simulation semantics 225
9.1 Supply network creation 225
9.2 Supply network simulation 227
9.3 Power state simulation 228
9.4 Power state transition detection 231
9.5 Simstate simulation 232
9.6 Transitioning from one simstate state to another 234
9.7 Simulation of retention 235
9.8 Simulation of isolation 241
9.9 Simulation of level-shifting 242
9.10 Simulation of repeaters 242

10. UPF information model 242
10.1 Overview 242
10.2 Components of UPF information model 243
10.3 Identifiers in information model (IDs) 244
10.4 Classification of objects 247
10.5 Example of design hierarchy 253
10.6 Object definitions 254

11. Information model application programmable interface (API) 313
11.1 Tcl interface 313
11.2 HDL interface 323

Annex A (informative) Bibliography 387

Annex B (normative) Value conversion tables 388
B.1 Overview 388
B.2 VHDL_SL2UPF 388
B.3 UPF2VHDL_SL 388
B.4 VHDL_SL2UPF_GNDZERO 388
B.5 UPF_GNDZERO2VHDL_SL 389
B.6 SV_LOGIC2UPF 389
B.7 UPF2SV_LOGIC 389
B.8 SV_LOGIC2UPF_GNDZERO 389
B.9 UPF_GNDZERO2SV_LOGIC 389
B.10 VHDL_TIED_HI 390
B.11 SV_TIED_HI 390
B.12 VHDL_TIED_LO 390
B.13 SV_TIED_LO 390

Annex C (informative) UPF query examples 391
C.1 Overview 391
C.2 Utility procs 391
C.3 High-level procs 392
C.4 Superseded UPF queries 394

Annex D (informative) Replacing deprecated and legacy commands and options 396
D.1 Overview 396
D.2 Deprecated and legacy constructs 396

Documents
Document Sharing Hub
Published by IEC under licence from IEEE. © 2018 IEEE. All rights reserved.
Full document details: BS IEC 61523-4:2023

D.3 Recommendations for replacing deprecated and legacy constructs 398

Annex E (informative) Low-power design methodology 401
E.1 Overview 401
E.2 Simple System on Chip (SoC) example design 401
E.3 Design, verification, and implementation flow 404
E.4 Power intent of the example design 407

Annex F (informative) Power-management cell definitions in UPF and Liberty 428
F.1 Introduction 428
F.2 define_always_on_cell 428
F.3 define_diode_clamp 430
F.4 define_isolation_cell 431
F.5 define_level_shifter_cell 434
F.6 define_power_switch_cell 436
F.7 define_retention_cell 438

Annex G (informative) Power-management cell modeling examples 442
G.1 Overview 442
G.2 Modeling always-on cells 442
G.3 Modeling cells with internal diodes 448
G.4 Modeling isolation cells 450
G.5 Modeling level-shifters 467
G.6 Modeling power-switch cells 484
G.7 Modeling state retention cells 494

Annex H (informative) IP power modeling for system-level design 506
H.1 Introduction 506
H.2 Overview of system-level IP power models 506
H.3 Content of system-level IP power models 507
H.4 Power calculation using power functions 508
H.5 Power model structure 510
H.6 Power model instantiation—example approach 511

Annex I (normative) Switching Activity Interchange Format 513
I.1 Syntactic conventions 514
I.2 Lexical conventions 515
I.3 Backward SAIF file 518
I.4 Library forward SAIF file 534
I.5 RTL forward SAIF file 542

Annex J (informative) Participants 547

Documents
Document Sharing Hub
Published by IEC under licence from IEEE. © 2018 IEEE. All rights reserved.
Full document details: BS IEC 61523-4:2023

DELAY AND POWER CALCULATION STANDARDS –

Part 4: Design and Verification of Low-Power, Energy-Aware Electronic Systems

FOREWORD

1) The International Electrotechnical Commission (IEC) is a worldwide organization for standardization comprising all national electrotechnical committees (IEC National Committees). The object of IEC is to promote international co-operation on all questions concerning standardization in the electrical and electronic fields. To this end and in addition to other activities, IEC publishes International Standards, Technical Specifications, Technical Reports, Publicly Available Specifications (PAS) and Guides (hereafter referred to as "IEC document(s)"). Their preparation is entrusted to technical committees; any IEC National Committee interested in the subject dealt with may participate in this preparatory work. International, governmental and non-governmental organizations liaising with the IEC also participate in this preparation.

   IEEE Standards documents are developed within IEEE Societies and Standards Coordinating Committees of the IEEE Standards Association (IEEE SA) Standards Board. IEEE develops its standards through a consensus development process, approved by the American National Standards Institute, which brings together volunteers representing varied viewpoints and interests to achieve the final product. Volunteers are not necessarily members of IEEE and serve without compensation. While IEEE administers the process and establishes rules to promote fairness in the consensus development process, IEEE does not independently evaluate, test, or verify the accuracy of any of the information contained in its standards. Use of IEEE Standards documents is wholly voluntary. *IEEE documents are made available for use subject to important notices and legal disclaimers (see* https://standards.ieee.org/ipr/disclaimers.html *for more information).*

   IEC collaborates closely with IEEE in accordance with conditions determined by agreement between the two organizations. This Dual Logo International Standard was jointly developed by the IEC and IEEE under the terms of that agreement.

2) The formal decisions of IEC on technical matters express, as nearly as possible, an international consensus of opinion on the relevant subjects since each technical committee has representation from all interested IEC National Committees. The formal decisions of IEEE on technical matters, once consensus within IEEE Societies and Standards Coordinating Committees has been reached, is determined by a balanced ballot of materially interested parties who indicate interest in reviewing the proposed standard. Final approval of the IEEE standards document is given by the IEEE Standards Association (IEEE SA) Standards Board.

3) IEC/IEEE Publications have the form of recommendations for international use and are accepted by IEC National Committees/IEEE Societies in that sense. While all reasonable efforts are made to ensure that the technical content of IEC/IEEE Publications is accurate, IEC or IEEE cannot be held responsible for the way in which they are used or for any misinterpretation by any end user.

4) In order to promote international uniformity, IEC National Committees undertake to apply IEC Publications (including IEC/IEEE Publications) transparently to the maximum extent possible in their national and regional publications. Any divergence between any IEC/IEEE Publication and the corresponding national or regional publication shall be clearly indicated in the latter.

5) IEC and IEEE do not provide any attestation of conformity. Independent certification bodies provide conformity assessment services and, in some areas, access to IEC marks of conformity. IEC and IEEE are not responsible for any services carried out by independent certification bodies.

6) All users should ensure that they have the latest edition of this publication.

7) No liability shall attach to IEC or IEEE or their directors, employees, servants or agents including individual experts and members of technical committees and IEC National Committees, or volunteers of IEEE Societies and the Standards Coordinating Committees of the IEEE Standards Association (IEEE SA) Standards Board, for any personal injury, property damage or other damage of any nature whatsoever, whether direct or indirect, or for costs (including legal fees) and expenses arising out of the publication, use of, or reliance upon, this IEC/IEEE Publication or any other IEC or IEEE Publications.

8) Attention is drawn to the normative references cited in this publication. Use of the referenced publications is indispensable for the correct application of this publication.

9) Attention is drawn to the possibility that implementation of this IEC/IEEE Publication may require use of material covered by patent rights. By publication of this standard, no position is taken with respect to the existence or validity of any patent rights in connection therewith. IEC or IEEE shall not be held responsible for identifying Essential Patent Claims for which a license may be required, for conducting inquiries into the legal validity or scope of Patent Claims or determining whether any licensing terms or conditions provided in connection with submission of a Letter of Assurance, if any, or in any licensing agreements are reasonable or non-discriminatory. Users of this standard are expressly advised that determination of the validity of any patent rights, and the risk of infringement of such rights, is entirely their own responsibility.

Published by IEC under licence from IEEE. © 2018 IEEE. All rights reserved.

Documents
Document Sharing Hub

Full document details: BS IEC 61523-4:2023

IEC 61523-4/IEEE Std 1801 was processed through IEC technical committee 91: Electronics assembly technology, under the IEC/IEEE Dual Logo Agreement. It is an International Standard.

The text of this International Standard is based on the following documents:

IEEE Std	FDIS	Report on voting
1801 (2018)	91/1870/FDIS	91/1884/RVD

Full information on the voting for its approval can be found in the report on voting indicated in the above table.

The language used for the development of this International Standard is English.

The IEC Technical Committee and IEEE Technical Committee have decided that the contents of this document will remain unchanged until the stability date indicated on the IEC website under webstore.iec.ch in the data related to the specific document. At this date, the document will be

- reconfirmed,
- withdrawn, or
- revised.

The contents of the corrigendum 1 (2024-02) have been included in this copy.

Published by IEC under licence from IEEE. © 2018 IEEE. All rights reserved.

Documents
Document Sharing Hub

Full document details: BS IEC 61523-4:2023

IEEE Standard for Design and Verification of Low-Power, Energy-Aware Electronic Systems

Sponsor

Design Automation Standards Committee
of the
IEEE Computer Society

Approved 27 September 2018

IEEE-SA Standards Board

Published by IEC under licence from IEEE. © 2018 IEEE. All rights reserved.

Documents
Document Sharing Hub

Full document details: BS IEC 61523-4:2023

Abstract: A method is provided for specifying power intent for an electronic design, for use in verification of the structure and behavior of the design in the context of a given power-management architecture, and for driving implementation of that power-management architecture. The method supports incremental refinement of power intent specifications required for IP-based design flows.

Keywords: corruption semantics, IEEE 1801™, interface specification, IP reuse, isolation, level-shifting, power-aware design, power domains, power intent, power modes, power states, progressive design refinement, retention, retention strategies

Documents
Document Sharing Hub
Published by IEC under licence from IEEE. © 2018 IEEE. All rights reserved.
Full document details: BS IEC 61523-4:2023

Important Notices and Disclaimers Concerning IEEE Standards Documents

IEEE documents are made available for use subject to important notices and legal disclaimers. These notices and disclaimers, or a reference to this page, appear in all standards and may be found under the heading "Important Notice" or "Important Notices and Disclaimers Concerning IEEE Standards Documents."

Notice and Disclaimer of Liability Concerning the Use of IEEE Standards Documents

IEEE Standards documents (standards, recommended practices, and guides), both full-use and trial-use, are developed within IEEE Societies and the Standards Coordinating Committees of the IEEE Standards Association ("IEEE-SA") Standards Board. IEEE ("the Institute") develops its standards through a consensus development process, approved by the American National Standards Institute ("ANSI"), which brings together volunteers representing varied viewpoints and interests to achieve the final product. Volunteers are not necessarily members of the Institute and participate without compensation from IEEE. While IEEE administers the process and establishes rules to promote fairness in the consensus development process, IEEE does not independently evaluate, test, or verify the accuracy of any of the information or the soundness of any judgments contained in its standards.

IEEE does not warrant or represent the accuracy or content of the material contained in its standards, and expressly disclaims all warranties (express, implied and statutory) not included in this or any other document relating to the standard, including, but not limited to, the warranties of: merchantability; fitness for a particular purpose; non-infringement; and quality, accuracy, effectiveness, currency, or completeness of material. In addition, IEEE disclaims any and all conditions relating to: results; and workmanlike effort. IEEE standards documents are supplied "AS IS" and "WITH ALL FAULTS."

Use of an IEEE standard is wholly voluntary. The existence of an IEEE standard does not imply that there are no other ways to produce, test, measure, purchase, market, or provide other goods and services related to the scope of the IEEE standard. Furthermore, the viewpoint expressed at the time a standard is approved and issued is subject to change brought about through developments in the state of the art and comments received from users of the standard.

In publishing and making its standards available, IEEE is not suggesting or rendering professional or other services for, or on behalf of, any person or entity nor is IEEE undertaking to perform any duty owed by any other person or entity to another. Any person utilizing any IEEE Standards document, should rely upon his or her own independent judgment in the exercise of reasonable care in any given circumstances or, as appropriate, seek the advice of a competent professional in determining the appropriateness of a given IEEE standard.

IN NO EVENT SHALL IEEE BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO: PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE PUBLICATION, USE OF, OR RELIANCE UPON ANY STANDARD, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE AND REGARDLESS OF WHETHER SUCH DAMAGE WAS FORESEEABLE.

Translations

The IEEE consensus development process involves the review of documents in English only. In the event that an IEEE standard is translated, only the English version published by IEEE should be considered the approved IEEE standard.

Documents
Document Sharing Hub

Published by IEC under licence from IEEE. © 2018 IEEE. All rights reserved.

Full document details: BS IEC 61523-4:2023

Official statements

A statement, written or oral, that is not processed in accordance with the IEEE-SA Standards Board Operations Manual shall not be considered or inferred to be the official position of IEEE or any of its committees and shall not be considered to be, or be relied upon as, a formal position of IEEE. At lectures, symposia, seminars, or educational courses, an individual presenting information on IEEE standards shall make it clear that his or her views should be considered the personal views of that individual rather than the formal position of IEEE.

Comments on standards

Comments for revision of IEEE Standards documents are welcome from any interested party, regardless of membership affiliation with IEEE. However, IEEE does not provide consulting information or advice pertaining to IEEE Standards documents. Suggestions for changes in documents should be in the form of a proposed change of text, together with appropriate supporting comments. Since IEEE standards represent a consensus of concerned interests, it is important that any responses to comments and questions also receive the concurrence of a balance of interests. For this reason, IEEE and the members of its societies and Standards Coordinating Committees are not able to provide an instant response to comments or questions except in those cases where the matter has previously been addressed. For the same reason, IEEE does not respond to interpretation requests. Any person who would like to participate in revisions to an IEEE standard is welcome to join the relevant IEEE working group.

Comments on standards should be submitted to the following address:

Secretary, IEEE-SA Standards Board
445 Hoes Lane
Piscataway, NJ 08854 USA

Laws and regulations

Users of IEEE Standards documents should consult all applicable laws and regulations. Compliance with the provisions of any IEEE Standards document does not imply compliance to any applicable regulatory requirements. Implementers of the standard are responsible for observing or referring to the applicable regulatory requirements. IEEE does not, by the publication of its standards, intend to urge action that is not in compliance with applicable laws, and these documents may not be construed as doing so.

Copyrights

IEEE draft and approved standards are copyrighted by IEEE under U.S. and international copyright laws. They are made available by IEEE and are adopted for a wide variety of both public and private uses. These include both use, by reference, in laws and regulations, and use in private self-regulation, standardization, and the promotion of engineering practices and methods. By making these documents available for use and adoption by public authorities and private users, IEEE does not waive any rights in copyright to the documents.

Photocopies

Subject to payment of the appropriate fee, IEEE will grant users a limited, non-exclusive license to photocopy portions of any individual standard for company or organizational internal use or individual, non-commercial use only. To arrange for payment of licensing fees, please contact Copyright Clearance Center, Customer Service, 222 Rosewood Drive, Danvers, MA 01923 USA; +1 978 750 8400. Permission to photocopy portions of any individual standard for educational classroom use can also be obtained through the Copyright Clearance Center.

Documents
Document Sharing Hub
Published by IEC under licence from IEEE. © 2018 IEEE. All rights reserved.
Full document details: BS IEC 61523-4:2023

Updating of IEEE Standards documents

Users of IEEE Standards documents should be aware that these documents may be superseded at any time by the issuance of new editions or may be amended from time to time through the issuance of amendments, corrigenda, or errata. An official IEEE document at any point in time consists of the current edition of the document together with any amendments, corrigenda, or errata then in effect.

Every IEEE standard is subjected to review at least every ten years. When a document is more than ten years old and has not undergone a revision process, it is reasonable to conclude that its contents, although still of some value, do not wholly reflect the present state of the art. Users are cautioned to check to determine that they have the latest edition of any IEEE standard.

In order to determine whether a given document is the current edition and whether it has been amended through the issuance of amendments, corrigenda, or errata, visit the IEEE-SA Website at http://ieeexplore.ieee.org/xpl/standards.jsp or contact IEEE at the address listed previously. For more information about the IEEE-SA or IEEE's standards development process, visit the IEEE-SA Website at http://standards.ieee.org.

Errata

Errata, if any, for all IEEE standards can be accessed on the IEEE-SA Website at the following URL: http://standards.ieee.org/findstds/errata/index.html. Users are encouraged to check this URL for errata periodically.

Patents

Attention is called to the possibility that implementation of this standard may require use of subject matter covered by patent rights. By publication of this standard, no position is taken by the IEEE with respect to the existence or validity of any patent rights in connection therewith. If a patent holder or patent applicant has filed a statement of assurance via an Accepted Letter of Assurance, then the statement is listed on the IEEE-SA Website at http://standards.ieee.org/about/sasb/patcom/patents.html. Letters of Assurance may indicate whether the Submitter is willing or unwilling to grant licenses under patent rights without compensation or under reasonable rates, with reasonable terms and conditions that are demonstrably free of any unfair discrimination to applicants desiring to obtain such licenses.

Essential Patent Claims may exist for which a Letter of Assurance has not been received. The IEEE is not responsible for identifying Essential Patent Claims for which a license may be required, for conducting inquiries into the legal validity or scope of Patents Claims, or determining whether any licensing terms or conditions provided in connection with submission of a Letter of Assurance, if any, or in any licensing agreements are reasonable or non-discriminatory. Users of this standard are expressly advised that determination of the validity of any patent rights, and the risk of infringement of such rights, is entirely their own responsibility. Further information may be obtained from the IEEE Standards Association.

Documents
Document Sharing Hub
Published by IEC under licence from IEEE. © 2018 IEEE. All rights reserved.
Full document details: BS IEC 61523-4:2023

IEEE Introduction

This introduction is not part of IEEE Std 1801-2018™, IEEE Standard for Design and Verification of Low-Power, Energy-Aware Electronic Systems.

The purpose of this standard is to provide portable, low-power design specifications that can be used with a variety of commercial products throughout an electronic system design, analysis, verification, and implementation flow.

When the electronic design automation (EDA) industry began creating standards for use in specifying, simulating, and implementing functional specifications of digital electronic circuits in the 1980s, the primary design constraint was the transistor area necessary to implement the required functionality in the prevailing process technology at that time. Power considerations were simple and easily assumed for the design as power consumption was not a major consideration and most chips operated on a single voltage for all functionality. Therefore, hardware description languages (HDLs) such as VHDL (IEC 61691-1-1/ IEEE Std 1076™[1]) and SystemVerilog (IEEE Std 1800™[2]) provided a rich set of capabilities necessary for capturing the functional specification of electronic systems, but no capabilities for capturing the power architecture (how each element of the system is to be powered).

As the process technology for manufacturing electronic circuits continued to advance, power (as a design constraint) continually increased in importance. Even above the 90 nm process node size, dynamic power consumption became an important design constraint as the functional size of designs increased power consumption at the same time battery-operated mobile systems, such as cell phones and laptop computers, became a significant driver of the electronics industry. Techniques for reducing dynamic power consumption—the amount of power consumed to transition a node from a `0` to `1` state or vice versa—became commonplace. Although these techniques affected the design methodology, the changes were relatively easy to accommodate within the existing HDL-based design flow, as these techniques were primarily focused on managing the clocking for the design (more clock domains operating at different frequencies and gating of clocks when logic in a clock domain is not needed for the active operational mode). Multi-voltage power-management methods were also developed. These methods did not directly impact the functionality of the design, requiring only level-shifters between different voltage domains. Multi-voltage power domains could be verified in existing design flows with additional, straightforward extensions to the methodology.

With process technologies below 90 nm, static power consumption has become a prominent and, in many cases, dominant design constraint. Due to the physics of the smaller process nodes, power is leaked from transistors even when the circuitry is quiescent (no toggling of nodes from `0` to `1` or vice versa). New design techniques have been developed to manage static power consumption. Power gating or power shut-off turns off power for a set of logic elements. Back-bias techniques are used to raise the voltage threshold at which a transistor can change its state. While back bias slows the performance of the transistor, it greatly reduces leakage. These techniques are often combined with multi-voltages and require additional functionality: power-management controllers, isolation cells that logically and/or electrically isolate a shutdown power domain from “powered-up” domains, level-shifters that translate signal voltages from one domain to another, and retention registers to facilitate fast transition from a power-off state to a power-on state for a domain.

The Unified Power Format (UPF) was developed to enable modeling of these new power-management techniques and to facilitate automation of design, verification, and implementation tools that must account for power-management aspects of a design. The initial version of UPF, developed by the Accellera Systems Initiative, focused primarily on modeling power distribution and its effects on the behavior of a system. In

[1] The IEEE standards or products referred to in this clause are trademarks of The Institute of Electrical and Electronics Engineers, Incorporated.
[2] IEEE publications are available from The Institute of Electrical and Electronics Engineers (http://standards.ieee.org/).

Published by IEC under licence from IEEE. © 2018 IEEE. All rights reserved.

Documents
Document Sharing Hub

Full document details: BS IEC 61523-4:2023

May 2007 that initial version was donated to the IEEE, and in March 2009 a new version, IEEE Std 1801, was released. That update of UPF added many new features, including the concept of successive refinement, more abstract modeling of system-power states, and more abstract modeling of supply networks.

This document, the latest revision of IEEE Std 1801, makes available further enhancements to UPF, including enhanced concepts for modeling power states and transitions at all levels of aggregation, enhanced support for methodologies such as successive refinement and bottom-up implementation, and a detailed information model that serves as the basis for enhanced package UPF functions and query functions. This current version also provides support for component power modeling for system-level power analysis in virtual prototyping applications.

Documents
Document Sharing Hub
Published by IEC under licence from IEEE. © 2018 IEEE. All rights reserved.
Full document details: BS IEC 61523-4:2023

IEEE Standard for Design and Verification of Low-Power, Energy-Aware Electronic Systems

1. Overview

1.1 Scope

This standard defines the syntax and semantics of a format used to express power intent in energy-aware electronic system design. *Power intent* includes the concepts and information required for specification and validation, implementation and verification, and modeling and analysis of power-managed electronic systems. This standard also defines the relationship between the power intent captured in this format and design intent captured via other formats (e.g., standard hardware description languages and cell libraries).

1.2 Purpose

The standard enables portability of power intent across a variety of commercial products throughout an electronic system design, analysis, verification, and implementation flow.

1.3 Key characteristics of the Unified Power Format

The Unified Power Format (UPF) provides the ability for electronic systems to be designed with power as a key consideration early in the process. UPF accomplishes this by allowing the specification of what was traditionally physical implementation-based power information early in the design process—at the register transfer level (RTL) or earlier. Figure 1 shows UPF supporting the entire design flow. UPF provides a consistent format to specify power-design information that may not be easily specifiable in a hardware description language (HDL) or when it is undesirable to directly specify the power semantics in an HDL, as doing so would tie the logic specification directly to a constrained power implementation. UPF specifies a set of HDL attributes and HDL packages to facilitate the expression of power intent in HDL when appropriate (see Table 4 and 11.2). UPF also defines consistent semantics across verification and implementation to check that what is implemented is the same as what has been verified.

Published by IEC under licence from IEEE. © 2018 IEEE. All rights reserved.

Documents
Document Sharing Hub

Full document details: BS IEC 61523-4:2023

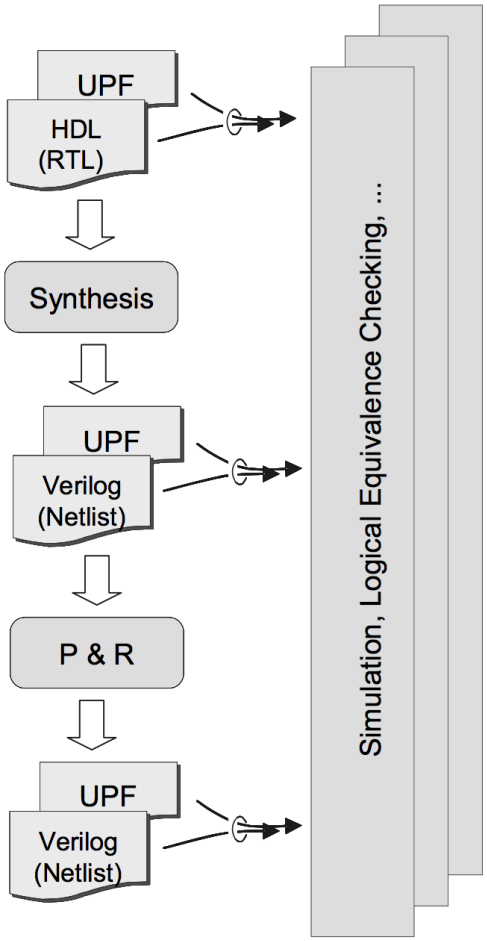

Figure 1—UPF tool flow

As indicated in Figure 1, UPF files are part of the design source and, when combined with the HDL, represent a complete design description: the HDL describing the logical intent and the UPF describing the power intent. Combined with the HDL, the UPF files are used to describe the intent of the designer. This collection of source files is the input to several tools, e.g., simulation tools, synthesis tools, and formal verification tools. UPF supports the successive refinement methodology (see 4.9) where power intent information grows along the design flow to provide needed information for each design stage.

— Simulation tools can read the HDL/UPF design input files and perform RTL power-aware simulation. At this stage, the UPF might only contain abstract models such as power domains and supply sets without the need to create the power and ground network and implementation details.

— A user may further refine the UPF specification to add implementation-related information. This further-refined specification may then be processed by synthesis tools to produce a netlist and optionally update the UPF fileset accordingly.

— In those cases where design object names change, a UPF file with the new names is needed. A UPF-aware logical equivalence checker can read the full design and UPF filesets and perform the checks to ensure power-aware equivalence.

— Place and route tools read both the netlist and the UPF files and produce a physical netlist, potentially including an output UPF file.

Published by IEC under licence from IEEE. © 2018 IEEE. All rights reserved.
Documents
Document Sharing Hub
Full document details: BS IEC 61523-4:2023

UPF is a concise, power intent specification capability. Power intent can be easily specified over many elements in the design. A UPF specification can be included with the other deliverables of intellectual property (IP) blocks and reused along with the other delivered IP. UPF supports various methodologies through carefully defined semantics, flexibility in specification, and, when needed, defined rational limitations that facilitate automation in verification and implementation.

1.4 Contents of this standard

The organization of the remainder of this standard is as follows:

— Clause 2 provides references to other applicable standards that are presumed or required for this standard.

— Clause 3 defines terms and acronyms used throughout the different specifications contained in this standard.

— Clause 4 describes the basic concepts underlying UPF.

— Clause 5 describes the language basics for UPF and its commands.

— Clause 6 details the syntax and semantics for each UPF power intent command.

— Clause 7 details the syntax and semantics for each UPF power-management cell command.

— Clause 8 defines a reference model for UPF command processing.

— Clause 9 defines simulation semantics for various UPF commands.

— Clause 10 defines the UPF information model.

— Clause 11 defines the UPF information model application programmable interface (API).

— Annex A lists potentially useful additional reference material.

— Annex B lists the predefined value conversion tables (VCTs) for use in power intent specifications.

— Annex C provides sample Tcl procs for retrieving power intent information.

— Annex D summarizes deprecated and legacy commands.

— Annex E provides an overview of UPF tool flows and use model with an illustrative example.

— Annex F provdes a summary of UPF power-management cell command semantics and Liberty mappings.

— Annex G provides examples of UPF power-management cell modeling.

— Annex H provides an overview of UPF use model for system-level IP power modeling.

— Annex I defines the Switching Activity Interchange Format (SAIF) for representing power-related activity in a design.

Documents
Document Sharing Hub
Published by IEC under licence from IEEE. © 2018 IEEE. All rights reserved.
Full document details: BS IEC 61523-4:2023

2. Normative references

The following referenced documents are indispensable for the application of this document (i.e., they must be understood and used, so each referenced document is cited in text and its relationship to this document is explained). For dated references, only the edition cited applies. For undated references, the latest edition of the referenced document (including any amendments or corrigenda) applies.

IEC 61691-1-1/IEEE Std 1076™, Behavioural languages—Part 1-1: VHDL Language Reference Manual.[10, 11, 12]

IEEE Std 1800™, IEEE Standard for SystemVerilog—Unified Hardware Design, Specification, and Verification Language.

ISO/IEC 19501:2005, Information technology—Open Distributed Processing—Unified Modeling Language (UML) Version 1.4.2.

3. Definitions, acronyms, and abbreviations

3.1 Definitions

For the purposes of this document, the following terms and definitions apply. The *IEEE Standards Dictionary Online* [B1][13] should be consulted for terms not defined in Clause 3.[14] Certain terms in this standard reflect their corresponding definitions in IEEE Std 1800™ or IEC 61691-1-1/IEEE Std 1076™, or they are listed in Annex A.

active component: A component that contains one or more input receivers and one or more output drivers whose values are functions of the inputs, but whose inputs and outputs are not directly connected; or any hardware description language (HDL) construct(s) that synthesize(s) to an active component.

active control signal: A control signal that is currently presenting the value (level) or transition (edge) that enables or triggers an active component to operate in a particular manner.

active power state: A power state whose logic expression—or, in certain cases, supply expression—evaluate to *True* at a given time.

activity: Any change in the value of a net, regardless of whether that change is propagated to an output.

analog port: A port that is part of a connection that delivers analog signals.

ancestor: Any **instance** between the current **scope** in the **logic hierarchy** and its **root scope**. When the current scope is a top-level module, it does not have any ancestors. *See also:* **descendant**.

anonymous object: An object that is not named in the context of Unified Power Format (UPF). Implementations may assign a legal name, but such names are not visible in the UPF context.

[10] IEC publications are available from the International Electrotechnical Commission (http://www.iec.ch) and the American National Standards Institute (http://www.ansi.org/).
[11] IEEE publications are available from The Institute of Electrical and Electronics Engineers (http://standards.ieee.org/).
[12] The IEEE standards or products referred to in this clause are trademarks of The Institute of Electrical and Electronics Engineers, Incorporated.
[13] The numbers in brackets correspond to those of the bibliography in Annex A.
[14] *IEEE Standards Dictionary Online* subscription is available at:
http://www.ieee.org/portal/innovate/products/standard/standards_dictionary.html.

Published by IEC under licence from IEEE. © 2018 IEEE. All rights reserved.
Documents
Document Sharing Hub
Full document details: BS IEC 61523-4:2023

available in a domain: A supply that can be used by implementation tools to power cells that they insert in the extent of the domain.

balloon latch: A retention element style in which a register's value is saved to a dedicated latch at power-down and the latch value is restored to the register at power-up.

boundary instance: An **instance** that has no parent or whose **parent** is in a different **power domain**.

child domain (of a **HighConn** port on the **lower boundary** of a **power domain**): The power domain whose **upper boundary** contains the corresponding **LowConn** of the **HighConn port**.

coarse grain switch: A power switch that is used to generate switched supply for a group of library cells. This is identified using the attribute `switch_cell_type: coarse_grain` in Liberty and design attribute **UPF_switch_cell_type coarse_grain** in Unified Power Format (UPF) (see Table 4).

component: A physical and logical construction that relates inputs to outputs.

composite domain: A power domain consisting of subordinate **power domains** called **subdomains**. All subdomains in a composite domain share the same primary supply set.

configuration UPF: A Unified Power Format (UPF) specification of the power-management configuration for a system.

connected: Attached together via a direct connection.

constraint UPF: A Unified Power Format (UPF) specification for an intellectual property (IP) block that defines constraints for any instance of this IP block that must be met by the power-management configuration of the system containing that instance.

correlated: A pair of **supply net**s or a pair of **supply set**s that are deemed to be at the same point in their voltage range when being considered for level-shifting. As such, when voltage levels are considered between them they should be mutually compared, e.g., minimum to minimum and maximum to maximum.

corruption semantics: The rules defining the behavior of logic in response to reduction or disconnection of power to that logic.

current scope: The design hierarchy location that serves as the immediate context for interpretation and execution of Unified Power Format (UPF) commands. Also, the **instance** specified by the **set_scope** command.

declared: Specified in the hardware description language (HDL) explicitly or implicitly via a Unified Power Format (UPF) command.

descendant: Any **instance** between the current scope in the **logic hierarchy** and its **leaf-level instance**s. *See also:* **ancestor**.

descendant subtree: A portion of a **logic hierarchy**, rooted at one **instance** in the hierarchy, and containing that **instance** and all of its **descendant**s.

design hierarchy: A hierarchical structure of nested definitions described in a hardware description language (HDL).

direct connection: A physical wire; or any hardware description language (HDL) construct(s) that synthesize(s) to a direct connection.

Published by IEC under licence from IEEE. © 2018 IEEE. All rights reserved.

Documents
Document Sharing Hub

Full document details: BS IEC 61523-4:2023

domain port: A **port** that is on the interface of a **power domain**.

driver: The source or drain of a transistor, if the drain or source is connected to a **power rail**; a complementary metal oxide semiconductor (CMOS) inverter that continually connects a node to power or ground; any **component** that sets the value of its output via a transistor or inverter; a constant assignment; any combinational logic including a buffer of any kind; any sequential logic; or any hardware description language (HDL) construct(s) that synthesize(s) to such combinational or sequential logic.

driver supply: For a **driver** that is a transistor, the supply connected to its source or drain; for a **driver** that is an inverter, the pair of supplies connected to the source/drain of the transistor pair comprising the inverter; or for an output of an **active component**, the related **supply set** of that output.

electrically equivalent: For **supply port**s/**net**s, connected (whether the connections are evident or not in the design) without any intervening switches, and therefore have the same value at all times from the perspective of any load; for **supply set**s/set handles, consisting of a set of electrically equivalent **supply net**s for each required function.

equivalent: A pair of **supply net**s, a pair of **supply set**s or a pair of **logic nets** that are considered to be identical for certain purposes. *See also:* **electrically equivalent; functionally equivalent**.

erroneous: A usage that is likely to lead to an error in the design, but that tools may not be able to detect and report.

extent (of a domain): The set of instances that comprise a **power domain**.

fanout domain (of a given port to which a given strategy applies): The **power domain** containing any of the following: receiving logic for that **port**, or a **leaf-level cell** instance **HighConn** input **port** that is connected to the given **port**, or a design top module **LowConn** output **port** that is connected to the given port.

feedthrough: A direct connection between two **port**s on the **interface of a power domain**, where the connection involves two **port**s on the upper boundary, or two **port**s on the lower boundary, or one of each; also, a direction connection between two **port**s of the same **leaf-level instance**.

feedthrough port: A **port** on the **interface of a power domain** that is part of a **feedthrough** through that domain, or a **port** on the interface of a **leaf-level instance** that is part of a **feedthrough** through that **instance**.

fine grain switch: A power switch that is used to generate switched supply for a single library cell. This is typically used to describe embedded macro power switches. This is identified using the attribute `switch_cell_type: fine_grain` in Liberty and design attribute **UPF_switch_cell_type fine_grain** in Unified Power Format (UPF) (see Table 4).

functionally equivalent: Functioning identically from the perspective of any load, either as a result of being **electrically equivalent**, or due to independent but parallel circuitry. Can be defined as interchangeable or non-interchangeable.

generate block: In the hardware description language (HDL) code, this represents a level of design hierarchy, although a generate block is not itself an **instance**. After synthesis, generate blocks do not exist as an independent level of hierarchy. It is illegal to create any Unified Power Format (UPF) objects in a **scope** that corresponds to a generate block.

golden source: The design together with the constraint Unified Power Format (UPF) and the configuration UPF.

Published by IEC under licence from IEEE. © 2018 IEEE. All rights reserved.
Documents
Document Sharing Hub
Full document details: BS IEC 61523-4:2023

hard macro: An instance that has **UPF_is_hard_macro** attribute set to **TRUE**. This instance typically represents a block that has been completely implemented and can be used as it is in other blocks. This can be modeled by an hardware description language (HDL) module for verification or as a library cell for implementation.

hierarchical name: A series of names separated by the **hierarchical separator character**, the final name of which is a legal hardware description language (HDL) name or Unified Power Format (UPF) name, and each preceding name is the name of an **instance** or **generate block** in which the following name is **declared**. *See also:* **hierarchical separator character**.

hierarchical separator character: A special character used in composing **hierarchical name**s. The hierarchical separator character is a slash (/).

HighConn: The side of a **port** connection that is higher in the **design hierarchy**; the actual signal associated with a formal **port** definition.

implementation UPF: The Unified Power Format (UPF) specification of how power distribution and control is to be implemented for a system.

inactive: A normally active component in a state in which it does not respond to **activity** on its inputs. Also, a control signal that is not currently presenting the value (level) or transition (edge) that enables or triggers an **active component** to operate in a particular manner.

instance: A particular occurrence of a SystemVerilog module (see IEEE Std 1800), very high speed integrated circuit (VHSIC) hardware description language (VHDL) entity (see IEC 61691-1-1/IEEE Std 1076), or library cell at a specific location within the design hierarchy.

interface of a power domain: The union of the upper boundary and the lower boundary of the **power domain**.

isolation: A technique used to provide defined behavior of a logic signal when its driving logic is not active.

isolation cell: An **instance** that passes logic values during normal mode operation and clamps its output to some specified logic value when a control signal is asserted.

leaf cell: *See:* **leaf-level cell**.

leaf instance: *See:* **leaf-level cell**.

leaf-level cell: An instance that has no descendants, or an instance that is a soft or hard macro.

leaf-level instance: *See:* **leaf-level cell**.

level-shifter: An **instance** that translates signal values from an input voltage swing to a different output voltage swing.

live slave: A retention element style in which the slave latch of a master-slave flip-flop (MSFF) is always on and therefore maintains the value of the MSFF during power-down.

logically equivalent: Logic **ports**/**nets** that are directly connected without any intervening logic and therefore have the same value at all times from the perspective of any sink.

logic hierarchy: An abstract view of a **design hierarchy** in which only those definitions representing **instance**s are included.

Documents
Document Sharing Hub
Published by IEC under licence from IEEE. © 2018 IEEE. All rights reserved.
Full document details: BS IEC 61523-4:2023

LowConn: The side of a **port** connection that is lower in the **design hierarchy**; the formal **port** definition.

lower boundary (of a power domain): The **HighConn** side of each **port** of each **boundary instance** in the **extent** of another **power domain** whose **parent** is in the **extent** of this domain.

macro: An **instance** that is a **soft macro** or a **hard macro.**

map: Identify a specific **model** corresponding to an abstract behavior. An **instance** of the **model** can then be used to implement the specific behavior.

model: A SystemVerilog module, very high speed integrated circuit (VHSIC) hardware description language (VHDL) entity/architecture, or Liberty cell.

named power state: A **power state** defined using **add_power_state** for a **supply set**, **power domain**, **composite domain**, group, **model**, or **instance**, including the **power state**s ON and OFF that are predefined for **supply set**s and the **power state**s UNDEFINED and ERROR that are predefined for all objects that have **power state**s.

net: The individual **net segment**s that make up a collection of interconnections between a collection of **port**s. A **net** may be named or anonymous.

net segment: A direct connection within a single **instance**.

nominal voltage: The designated voltage value of a supply object. The actual voltage presented at the transistor may be different from the **nominal voltage**.

parent: The immediate **ancestor** of a given **instance** within the **logic hierarchy**.

parent domain (of a LowConn port on the upper boundary of a power domain): The **power domain** whose **lower boundary** contains the corresponding **HighConn** of the **LowConn** port.

passive component: A direct connection; a **component** that has neither a **receiver** nor a **driver**, whose output is connected to its input, and therefore its output is always the same as its input, e.g., a pass transistor; or any hardware description language (HDL) construct(s) that synthesize(s) to a **feedthrough** component.

path: A direct connection between a **driver** and a **receiver**. A **path** can be identified by its driver and/or receiver supplies. A **port** may exist on a single path or multiple paths. Multiple ports can exist on the same path. A port may be present on multiple paths, connected to the same set of driver/receiver supplies.

pg_type: An attribute of a port that indicates its use in providing power to a cell.

port: A **connection** on the interface of a SystemVerilog module or very high speed integrated circuit (VHSIC) hardware description language (VHDL) entity. Also, a **port** on the **interface of a power domain**.

power domain: A collection of **instance**s that are treated as a group for power-management purposes. The **instance**s of a power domain typically, but do not always, share a **primary supply set**. A power domain can also have additional supplies, including **retention** and **isolation** supplies.

power rail: The physical implementation of a power **supply net**.

power state: A subset of the functional states of an object that have the same characteristics with respect to power supply (for a **supply set**) or power consumption (for a **power domain**, **composite domain**, group, **model**, or **instance**).

Documents
Document Sharing Hub
Published by IEC under licence from IEEE. © 2018 IEEE. All rights reserved.
Full document details: BS IEC 61523-4:2023

power state table (PST): A table that specifies the legal combinations of **supply states** for a set of supply objects (**supply ports**, **supply net**s, and/or **supply set functions**).

primary supply set: The **supply net** connections inferred for all **instance**s in the **power domain**, unless overridden.

receiver: The gate of a transistor; the input to an inverter; any **component** whose behavior is determined by an input signal; any combinational logic including a buffer of any kind; any sequential logic; or any hardware description language (HDL) construct(s) that synthesize(s) to such combinational or sequential logic.

receiver supply: For a **receiver** that is the gate of a transistor, the supply connected to that transistor's source or drain; for a **receiver** that is the input to an inverter, the pair of supplies connected to the source/drain of the transistor pair comprising the inverter; or for a **receiver** that is part of an **active component**, the primary supply of the **power domain** to which that **receiver** belongs or, in some cases, the secondary supply of the **component** if it has a secondary supply.

regulator: An **instance** that takes a set of input **supply net**s and provides the source for a set of output **supply net**s. The output voltage is a function of the input voltages and the logical state of any control signals.

retention: Enhanced functionality associated with selected **sequential element**s or a memory such that memory values can be preserved during the power-down state of the primary supplies.

retention register: A register that extends the functionality of a **sequential element** with the ability to retain its memory value during the power-down state.

rooted name: The **hierarchical name**, relative to the **current scope**, of an object in the **logic hierarchy** or a Unified Power Format (UPF) object defined for a **scope** in the hierarchy.

root scope: The topmost scope in the **logic hierarchy**, which contains an implicit instance of each top-level module.

root supply driver: The origin of a supply, e.g., an on-system voltage regulator, bias generator modeled in hardware description language (HDL), or an off-chip supply source; also, any supply object that functions as a **root supply driver**, including a primary supply input to the design, a **leaf-level instance** supply output port, a power switch output port, and any supply object that is an input to a resolved **supply net**. *See also*: **supply source**.

self domain (of a **port** to which a given strategy applies): **The power domain** for which the strategy has been defined.

scope: A region in which names may be defined; such a region is either an hardware description language (HDL) model or **instance** in the **logic hierarchy** or a Unified Power Format–defined global context, **power state table**, **supply set**, **power domain**, **composite domain**, group, or **strategy**.

silicon UPF: *See:* **implementation UPF** (Unified Power Format).

simple name: An identifier that denotes an object declared in a given **scope** and is not a **hierarchical name**.

simstate: The level of operational capability supported by a given **power state** of a **supply set**.

sink: A **receiver**; the **HighConn** of an input port or inout port of an **instance**; or the **LowConn** of an output port or inout port of an **instance**.

Published by IEC under licence from IEEE. © 2018 IEEE. All rights reserved.

Documents
Document Sharing Hub

Full document details: BS IEC 61523-4:2023

soft macro: An **instance** that has **UPF_is_soft_macro** attribute set to **TRUE**. This instance typically is represented by the original register transfer level (RTL) and Unified Power Format (UPF) from which its implementation will be derived.

source: A **driver**; the **LowConn** of an input port or inout port of an **instance**; or the **HighConn** of an output port or inout port of an **instance**.

state element: A sequential element such as a flip-flop, latch, or memory element. Also, a conditionally stored value in register transfer level (RTL) code from which a sequential element would be inferred.

strategy: A rule that specifies where and how to apply **isolation**, level-shifting, state retention, and buffering in the implementation of power intent.

subdomain: A member of the set of domains comprising a composite **power domain**.

supply function: An abstraction of a **supply net** in a **supply set**, the name of which identifies the purpose of the corresponding **net** in the **supply set**.

supply net: An hardware description language (HDL) representation of a **power rail**.

supply port: A connection point for **supply net**s.

supply set: A collection of **supply function**s that in aggregate provide a complete power source.

supply source: A **supply port** that propagates but does not originate a supply value.

supply subnet: A set of **electrically equivalent supply port**s, **supply net**s, and/or **supply set** functions.

switch: An **instance** that conditionally connects one or more input **supply net**s to a single output **supply net** according to the logical state of one or more control inputs.

terminal boundary: The boundary formed around a **soft** or a **hard macro**.

top-level instance: An implicit **instance** corresponding to a top-level module.

upper boundary (of a power domain): The **LowConn** side of each **port** of each **boundary instance** in the **extent** of this **power domain**.

3.2 Acronyms and abbreviations

CMOS	complementary metal oxide semiconductor
DFT	Design for Test
EDA	electronic design automation
HDL	hardware description language
IP	intellectual property
MSFF	master-slave flip-flop
NMOS	N-channel metal oxide semiconductor

Published by IEC under licence from IEEE. © 2018 IEEE. All rights reserved.
Documents
Document Sharing Hub
Full document details: BS IEC 61523-4:2023

PG	power/ground
PMOS	P-channel metal oxide semiconductor
PST	power state table
ROM	read-only memory
RTL	register transfer level
SAIF	Switching Activity Interchange Format
SoC	System on Chip
Tcl	Tool Command Language
UPF	Unified Power Format
VCT	value conversion table
VHDL	very high speed integrated circuit (VHSIC) hardware description language
VHSIC	Very High Speed Integrated Circuit

4. Concepts

4.1 Introduction

Clause 4 provides an overview of concepts involved in defining power intent using Unified Power Format (UPF). These concepts include those related to the representation of the design structure and functionality in one or more hardware description languages (HDLs), as well as those related to power-management structures and functionality defined for and/or added to the design to model intended power-management capabilities.

The structure and functionality of a design is specified using HDLs such as SystemVerilog or very high speed integrated circuit (VHSIC) hardware description language (VHDL). Each HDL has specific terminology and concepts that are unique to that language, but all HDLs share some common concepts and capabilities. A typical design may be expressed in one or more HDLs.

UPF is defined in terms of a generalized abstraction of an HDL-based design hierarchy. This abstraction enables the UPF definition to apply to a design expressed in any of the three HDLs previously mentioned, or in any combination thereof, while at the same time minimizing the complexity of the UPF definition. Clause 4 presents the abstract model and maps it to specific HDL concepts.

UPF is intended to apply to a design as its representation changes from an abstract functional model to a concrete physical model, during which process the power intent expressed in UPF becomes realized as part of the implementation. Because of this, the abstract logic hierarchy that is the basis of the UPF definition shall be understood in terms of both functional specification and physical implementation.

Published by IEC under licence from IEEE. © 2018 IEEE. All rights reserved.
Documents
Document Sharing Hub
Full document details: BS IEC 61523-4:2023

4.2 Design structure

4.2.1 Transistors

At the lowest level, UPF focuses on controlling power (or more precisely, voltage and current) delivered to transistors. These are usually assumed to be digital complementary metal oxide semiconductor (CMOS) transistors, but they could be analog devices as well, or implemented in other technologies. The gate connection of a transistor is a receiver; the source of the signal provided to a gate (in CMOS, typically the output of a `P/N` transistor pair) is a driver.

4.2.2 Standard cells

Transistors are seldom modeled individually in an HDL description; typically, collections of transistors are represented by standard cells that have been developed as part of a particular technology library, which is usually expressed in the Liberty library format (see [B4]). Such cells typically have a primary supply (power and ground) and can also have a secondary supply for related behavior (e.g., state retention).

Power-management cells defined in UPF (see Clause 7) are also considered to be standard cells.

4.2.3 Hard macros

A library can also contain hard macros, which provide predefined physical implementations for much larger and more complex functions. A hard macro can have multiple supplies.

4.3 Design representation

4.3.1 Models

Library elements have corresponding behavioral models for use in simulation. These models may or may not include power and ground pins for their supplies. Standard cell models are usually written as SystemVerilog modules and use constructs such as SystemVerilog built-in primitives or user-defined primitives (UDPs) to express the relatively simple behavior of a standard cell. They can also be written as VHDL design entities (entity/architecture pairs) using package `VITAL`, which provides SystemVerilog-like primitive modeling capabilities. Hard macro models can be written in either language, using more complex behavioral constructs such as SystemVerilog initial blocks and always blocks, or VHDL processes and concurrent statements.

4.3.2 Netlist

A netlist is a collection of unique instances of standard cells and hard macros, interconnected by nets (SystemVerilog) or signals (VHDL). Such instances are considered to be leaf-level instances because their models are not constructed from an interconnection of subordinate instances, but instead are built using behavioral or functional HDL statements. A netlist can also include hierarchical instances, i.e., instances of a model that is itself defined as a netlist.

A power/ground (PG) netlist is a netlist containing cell and/or hard macro instances that include power and ground pins and a representation of the power and ground supply routing for those instances. A non-PG netlist is one that does not include any representation of the power supply network.

Published by IEC under licence from IEEE. © 2018 IEEE. All rights reserved.
Documents
Document Sharing Hub
Full document details: BS IEC 61523-4:2023

4.3.3 Synthesizable models

A synthesizable model is written using the synthesizable subset of SystemVerilog or VHDL and can be mapped to a functionally equivalent netlist by a logic synthesis tool. Synthesis involves identifying or inferring the state elements needed to implement the specified behavior and implementing the combinational logic interconnecting those elements and the model's ports.

For many synthesizable HDL constructs, synthesis creates combinational or sequential logic elements that are ultimately defined in terms of transistors, which in turn define drivers and receivers. In particular, any synthesizable statement that involves conditional computation or conditional updating of an output will most likely create logic. In contrast, unconditional assignment statements and port associations typically result in interconnect, not logic; for such HDL constructs, no drivers or receivers are created. In particular, ports do not create drivers; it is the logic driving a port that creates a driver for the port and for the net associated with the port.

4.3.4 HDL scopes

An HDL model defines one or more scopes. A scope is a region of HDL text within which names can be defined. Such names are typically visible (i.e., can be referenced) within the scope in which they are defined and, in certain cases, in other scopes (e.g., nested scopes). A SystemVerilog model usually defines a single scope for the whole model. A VHDL model often defines multiple scopes; one for the whole model, plus other nested scopes for process statements and block statements. `generate` statements in either HDL are also considered to be nested scopes within the model's top-level scope.

4.3.5 Design hierarchy

A design hierarchy is constructed by defining one model in terms of interconnected instances of other models. Each instance represents a subtree of the hierarchy; the boundary between this subtree and its parent instance is defined by the interface of the model that has been instantiated to create the subtree. The interface consists of the model's ports, together with the nets associated with those ports for the instance that created this subtree. In SystemVerilog, a port is defined as having two sides: a *HighConn* and a *LowConn*. The LowConn represents the port declaration in the model; the HighConn represents an instance of that port associated with an instance of the model, and therefore indirectly the net attached to that port instance. In VHDL, a somewhat different distinction is made between a formal port of a model and the actual signal associated with that port for a given instance of the model. In the context of UPF, regardless of what HDL is involved, the term *LowConn* means the (formal) port declaration in the model definition, and the term *HighConn* means the port of an instance of a model and by extension the net or signal connected to that port.

An HDL model that is not instantiated in any other instance is a top model, or simply `top`. A given design hierarchy usually contains a single top, but it may contain multiple tops in certain cases (e.g., if the design and the testbench in a simulation are modeled separately—neither instantiates the other). Each top is considered to be implicitly instantiated within the *root scope*. In SystemVerilog, the root scope is `$root`; in VHDL, the root scope is the *root declarative region*. The instance name of such an implicit instance is the same as the model name.

4.3.6 Logic hierarchy

UPF assumes a somewhat more abstract model of the design hierarchy. This abstract model is called the *logic hierarchy*. As usual, the topmost scope is still the root scope and modules that are not instantiated elsewhere are the top modules (and instances) of the hierarchy. However, in the logic hierarchy, each scope corresponds to a whole instance; internal scopes presented in the design hierarchy are not modeled. In

Published by IEC under licence from IEEE. © 2018 IEEE. All rights reserved.
Documents
Document Sharing Hub
Full document details: BS IEC 61523-4:2023

particular, HDL `generate` statements, which are considered to be internal scopes in the respective language definitions, are assumed to be collapsed into the parent module scope in the logic hierarchy.

UPF generally allows references to the names of objects defined anywhere in the subtree descending from a given instance when the *current scope* is set to that instance. Such references are called *rooted names*, meaning they are hierarchical names relative to the current scope. If the design hierarchy contains `generate` statements that have been collapsed in the logic hierarchy, then the hierarchical name of an object in the logic hierarchy may include simple names that encode the collapsed scope names.

UPF also uses the logic hierarchy as a framework for locating the power-management objects used to represent power-management concepts, e.g., power domains and power state tables (PSTs). Each such object is effectively declared in a specific scope of the logic hierarchy, and the name of the scope can be used as the prefix of the name of the object. Furthermore, certain UPF-defined objects act as scopes themselves, in which other subordinate named objects can be defined. In such cases, the name of the UPF-defined parent object can be used as the prefix of the name of any of its subordinate objects.

The logic hierarchy can be viewed as a purely conceptual structure that is independent of the eventual physical implementation. Alternatively, the logic hierarchy can be viewed as an indication of the floor plan to be used in the physical implementation. Either view can be used, but it is best to adopt one view or the other for a given design because the choice can affect how the power intent is expressed in UPF.

4.3.7 Hierarchy navigation

In UPF, commands are executed in the context of a scope within the logic hierarchy. The **set_scope** command (see 6.52) is used to navigate within the hierarchy and to set the current scope within which commands are executed.

Consistent with SystemVerilog `$root`, the root of the logic hierarchy is the scope in which the top modules are implicitly instantiated. Other locations within the logic hierarchy are referred to as the *design top instance*, which has a corresponding *design top module*, and the current scope.

The design top instance and design top module are typically paired: the design top instance (represented by a hierarchical name relative to the root scope) is an instance in the hierarchy representing a design for which power intent has been defined, and the design top module is the module for which the UPF file expressing this power intent has been written. The association between the UPF file and the design top module is specified in the UPF file using **set_design_top** (see 6.42); this UPF file is then typically applied to each instance of that module in a larger system.

The current scope is an instance that is, or is a descendant of, the design top instance (represented by a relative pathname from the design top instance).

The **set_scope** command (see 6.52) changes the current scope locally within the subtree depending on the current design top instance/module. Since the design top instance is typically an instance of the design top module, they both have the same hierarchical substructure; therefore, **set_scope** can be written relative to the module, but still work correctly when applied to an instance. The **set_scope** command is only allowed to change scope within this subtree. It cannot change the scope to a scope above the design top instance or to a scope that is, or is below, a leaf-level instance.

The design top instance and design top module are initially set by the tool, possibly with direction from the user. They are implicitly changed when **load_upf** is invoked with the **-scope** argument or when **apply_power_model** is invoked to apply a power model to a given instance.

Published by IEC under licence from IEEE. © 2018 IEEE. All rights reserved.
Documents
Document Sharing Hub
Full document details: BS IEC 61523-4:2023

4.3.8 Ports and nets

Ports define connection points between adjacent levels of hierarchy. In HDL, ports are defined as part of the interface of a module and therefore exist for each instance of the module. Nets define interconnections between a collection of ports. In HDL, nets are defined within a module and therefore exist within each instance of the module.

A port has two sides. The top side is the HighConn side, which is visible to the parent of the instance whose interface contains the port. The bottom side is the LowConn side, which is visible internal to the instance whose interface contains the port.

When a net in the current scope is connected to a port on a child instance, the connection is made to the HighConn side of the port. When a net in the current scope is connected to a port defined on the interface of the instance that is the current scope, the connection is made to the LowConn side of the port.

A port can be referenced wherever a net is required. Such a reference refers to the LowConn side of the port. A port can be thought of as being implicitly connected to an implicit net created with the same name and in the same scope as the LowConn side of the port.

The implicit net, created when a port is created, cannot be a resolved net. Therefore, it shall be an error for this implicit net to be connected to multiple drivers.

4.3.9 Connecting nets to ports

In an HDL description, ports are typically required to pass nets from one level of hierarchy to another. In UPF, a net in the current scope can be connected to the LowConn of any port declared in the same scope or to the HighConn of any port within its descendant subtree. If the port is not declared in the same scope as the net, additional ports, nets, and port/net associations may be created to establish the connection from the net to the port. Such implicitly created ports and nets shall have the same simple name as the net being connected unless that name conflicts with the name of an existing port or net; in which case, to avoid a name conflict, the tool shall create a name that is unique for that scope.

NOTE—Nets are propagated as necessary through the descendant subtree and may be renamed to avoid name collision; therefore, the same simple name in different scopes may refer to nets that are independent and unconnected.[15]

Implicitly created ports and nets should not be referenced directly by UPF commands, since the names of such ports and nets may not be the same as the original net name. These implicitly created ports and nets are merely a method of implementing a UPF connection in terms of valid HDL connections, when the UPF-specified power intent is represented in HDL form.

4.3.10 Representing SystemC design for power analysis

IEEE Std 1801™-2015 supports power analysis in system-level design; see Annex H for more details. In addition to the existing HDL support (SystemVerilog, VHDL), IEEE Std 1801-2015 system-level IP power models can also be applied to design descriptions that are written in SystemC.

Existing UPF commands like **set_scope**, **create_power_domain**, **apply_power_mode**, etc. can be applied to a SystemC design in the same way as existing HDL support.

[15] Notes in text, tables, and figures of a standard are given for information only and do not contain requirements needed to implement this standard.

Published by IEC under licence from IEEE. © 2018 IEEE. All rights reserved.

Documents
Document Sharing Hub

Full document details: BS IEC 61523-4:2023

A design description in SystemC is treated in the same way as other HDLs (scopes, design hierarchy, etc.) and SystemC generally follows the same rules of other HDLs.

4.4 Power architecture

4.4.1 Introduction

A UPF power intent specification defines the power architecture to be used in managing power distribution within a given design. The power architecture defines how the design is to be partitioned into regions that have independent power supplies, and how the interfaces between, and interaction among, those regions will be managed and mediated.

4.4.2 Power domains

A power domain is a collection of instances that are typically powered in the same way. In the physical implementation, the instances of a power domain are typically placed together and powered by the same power rails. In the logic hierarchy, the instances of a power domain are typically part of the same subtree of the hierarchy, or of sibling subtrees with a common ancestor, and powered by the same supply nets.

A power domain is defined within a scope (or instance) in the logic hierarchy. The definition of the power domain identifies the uppermost instances of the domain: those that define the upper boundary of the domain. For any given instance included in the power domain, a child instance of the given instance is transitively included in the power domain, unless that child instance is explicitly excluded from this power domain or is explicitly included in the definition of another power domain.

More formally, a boundary instance of a given power domain is any instance that has no parent (it is an implicit instance of a top-level module) or whose parent is in the extent of a different power domain. It is possible for one boundary instance of a power domain to be an ancestor of another boundary instance of the same power domain. This occurs when one instance is in the extent of a given power domain and both an ancestor and a descendant of that instance are in the extent of a second power domain. In this case, both the ancestor and the descendant may be boundary instances of the second domain. A domain with such a structure is referred to as a *donut* power domain.

The upper boundary of a power domain consists of the LowConn side of each port on each boundary instance in the domain. The lower boundary of a domain consists of the HighConn side of each port on each child instance that is in some other power domain. Both boundaries include any logic ports added to the design for power management. The interface of a power domain consists of the upper boundary and the lower boundary.

The instance in the logic hierarchy in which a power domain is defined is called the *scope* of the power domain. The set of instances that belong to a power domain are said to be the extent of that power domain. This distinction is important: while a given instance can be the scope of multiple power domains, it can be in the extent of one and only one power domain. As a consequence of these definitions, all instances within the extent of a domain are necessarily within the scope of the domain or its descendants.

A power domain can be either contiguous or non-contiguous. In the physical implementation, a contiguous power domain is one in which all instances are placed together; a non-contiguous power domain is one in which instances in the domain are placed in two or more disjoint locations. A power domain is contiguous within the logic hierarchy if it contains a single boundary instance; it is non-contiguous within the logic hierarchy if it includes multiple boundary instances.

Published by IEC under licence from IEEE. © 2018 IEEE. All rights reserved.
Documents
Document Sharing Hub
Full document details: BS IEC 61523-4:2023

For a non-contiguous power domain, a connection from an instance in the extent of the power domain to some other instance in the extent of the domain may need to be routed through another power domain.

Power domains that share a primary supply set can be composed together to form a larger power domain such that operations performed on this larger power domain apply transitively to each subdomain. In this way, unnecessary power domains may be aggregated together and handled as one for simplicity.

After UPF-specified power intent has been completely applied, it shall be an error if any instance is not included in a power domain.

4.4.3 Drivers, receivers, sources, and sinks

A logic signal in the design originates at an active component (the driver) and terminates at another active component (the receiver). Along the way it may pass through ports and nets. The driver and any port it passes through on the way to a receiver is considered a source; the receiver and any port it passes through on the way from the driver is considered a sink. For example, a buffer defines both a source and a sink: the buffer's output port is a source; the buffer's input port is a sink.

A signal traversing a power domain may or may not be driven within the power domain. A port is neither a driver nor a receiver; it merely propagates a signal across a hierarchy boundary. If a port on the interface of a power domain is connected directly to another port on the interface of the same power domain, without going through an active component, the connection between those two ports has neither a driver nor a receiver in that domain. In this case, the connection is a feedthrough path through that domain.

HDL assignment statements may include delays, which either represent inertial delay (resulting from transistor switching) or transport delay (resulting from propagation along a wire). However, synthesis tools typically ignore such delays; therefore, the inclusion of such a delay, whether inertial or transport, does not by itself imply that an active component will be inferred from the assignment. For this reason, delays are not considered to create drivers or receivers.

A connection may be thought to exist in a given domain, if a user so chooses, but since a connection is by definition a passive component, it has no driver in the domain in which it exists and therefore is not affected or corrupted by the power state of the domain in which it exists.

4.4.4 Isolation and level-shifting

Two power domains interact if one contains logic that is the driver of a net and the other contains logic that is a receiver of the same net. When both power domains are powered up, the receiving logic should always see the driving logic's output as an unambiguous `1` or `0` value, except for a very short time when the value is in transition. The structure of CMOS logic typically means that minimal current flow will occur when the input value to a gate is a `1` or `0`. However, if the driving logic is powered down, the input to the receiving logic can float between `1` or `0`. This can cause significant current to flow through the receiving logic, which can damage the circuit. An undriven input can also cause functional problems if it floats to an unintended logic value.

To avoid this problem, isolation cells are inserted at the boundary of a power domain such that the receiving logic always sees an unambiguous `1` or `0` value. Isolation may be inserted for an input or for an output of the power domain. An isolation cell operates in two modes: normal mode, in which it acts like a buffer, and isolation mode, in which it clamps its output to a defined value. An isolation enable signal determines the operational mode of an isolation cell at any given time.

Two interacting power domains may also be operating with different voltage ranges. In this case, a logic `1` value might be represented in the driving domain using a voltage that would not be seen as an unambiguous

Published by IEC under licence from IEEE. © 2018 IEEE. All rights reserved.
Documents
Document Sharing Hub
Full document details: BS IEC 61523-4:2023

1 in the receiving domain. Level-shifters are inserted at a domain boundary to translate from a lower to a higher voltage, and sometimes from a higher to a lower voltage as well. The translation means that the logic value sent by the driving logic in one domain is correctly received by the receiving logic in the other domain.

Isolation and level-shifting are often implemented in combination, so one standard cell implements both functions. UPF includes support for such "combo" cells.

Isolation and level-shifter strategies specify that isolation and level-shifter cells are to be inserted in specified locations. However, there are some cases where implementation tools may choose not to insert such cells, or to optimize redundant insertion of such cells. For example, isolation/level-shifters on floating ports that appear to have no drivers, or have constant drivers, may be removed or transformed, provided the resulting behavior is unchanged. To prevent implementation tools from applying such optimizations, isolation and level-shifting strategies can instead specify that the respective cells are to be inserted regardless of optimization possibilities.

4.4.5 State retention

State retention is the ability to retain the value of a state element in a power domain while switching off the primary power to that element, and being able to use the retained value as the functional value of the state element upon power-up. State retention can enable a power domain to return to operational mode more quickly after a power-down/power-up sequence and it can be used to maintain state values that cannot be easily recomputed on power-up. State retention can be implemented using retention memories or retention registers. Retention registers are sequential elements (latches or flip-flops) that have state retention capability.

For a retention register, the following terms apply:

— *Register value* is the data held in the storage element of the register. In functional mode, this value gets updated on the rising/falling edge of clock or gets set or cleared by set/reset signals, respectively.

— *Retained value* is the data in the retention element of retention register. The retention element is powered by the retention supply.

— *Output value* is the value on the output of the register.

Depending on how the retained value is stored and retrieved, there are at least two flavors of retention registers, as follows:

a) *Balloon-style retention*: In a balloon-style retention register, the retained value is held in an additional latch, often called the *balloon latch*. In this case, the balloon element is not in the functional data-path of the register.

b) *Master/slave-alive retention*: In a master/slave-alive retention register, the retained value is held in the master or slave latch. In this case, the retention element is in the functional data-path of the register.

A balloon-style retention register typically has additional controls to transfer data from a storage element to the balloon latch, also called the *save step*, and transfer data from the balloon latch to the storage element, also called the *restore step*. The ports to control the save/restore pins of the balloon-style retention register need to be available in the design to describe and implement this style of registers.

A master/slave-alive retention register typically does not have additional save/restore controls as the storage element is the same as the retention element. Additional control(s) on the register may park the register into a quiescent state and protect some of the internal circuitry during power-down state, and thus

Published by IEC under licence from IEEE. © 2018 IEEE. All rights reserved.
Documents
Document Sharing Hub
Full document details: BS IEC 61523-4:2023

the retention state is maintained. The restore in such registers typically happens upon power-up, again owing to the storage element being the same as the retention element. Thus, this style of registers may not specify save/restore signals, but may specify a retention condition that could take the register in and out of retention.

4.5 Power distribution

4.5.1 Overview

The electric current transported by a supply net originates at a root supply driver, which can be an on-chip voltage regulator, an embedded power switch, a bias generator, or an off-chip supply source. A power switch output and a resolved supply net are both also considered to be root supply drivers for semantic consistency.

Each supply subnet (see 3.1) has an associated root supply driver. If the supply subnet includes a primary supply input, the root supply driver is an implicit driver representing an off-chip supply source. If the supply subnet includes a hard macro cell output supply port, the root supply driver is an implicit driver representing an embedded power switch or supply regulator. If the supply subnet includes a switch output port, the root supply driver is the power switch output. If the supply subnet includes one or more resolved supply nets, the root supply driver is the output of the common resolution function shared by those resolved supply nets.

Initially, the root supply driver drives the supply subnet with the value `{OFF, unspecified}`. The package UPF functions `supply_on()` and `supply_off()` may be called to change the driving value of the root supply driver that drives a given supply subnet. These functions may be applied to any supply object in the supply subnet, provided that distribution of the supply value (or the result of a resolution of this supply value and other supply values) to loads of the supply network does not require violating the directionality of any port in the supply subnet.

A supply net can have one or more supply sources, depending upon its resolution type. During UPF processing, if the number of sources connected to a supply net do not conform to the requirements of its resolution type, an error shall be reported. At any given time during simulation, if the sources of a supply net do not conform to the requirements of its resolution type, the resolved value of the supply net at that time is set to `{UNDETERMINED, unspecified}`.

A power switch can have one or more input supply ports and one output supply port. Each input supply port can have one or more state definitions. At any given time during simulation, if the state definitions of a given input supply port are contradictory, or if multiple incompatible inputs are enabled at the same time, or if any input supply port is in an error state, the resolved value of the output supply port at that time is set to `{UNDETERMINED, unspecified}`.

The semantics defined in this standard, such as the supply net resolution functions, presume an idealized supply network with no voltage drop; the semantics for supply network resolution with modeled-voltage drop are outside the scope of this standard.

4.5.2 Supply network elements

Supply network objects (supply ports, supply nets, and switches) are created within the logic hierarchy to provide connection points for a root supply and to propagate the value of a root supply throughout a portion of the design. Supply network objects are created independent of power-domain definitions. This allows sharing of common components of the supply distribution network across multiple power domains.

Published by IEC under licence from IEEE. © 2018 IEEE. All rights reserved.

Documents
Document Sharing Hub

Full document details: BS IEC 61523-4:2023

4.5.3 Supply ports and nets

4.5.3.1 Introduction

Supply ports provide a connection point for supply nets where they cross a hierarchy boundary. Supply nets can be used to create a connection between two supply ports or from a supply port to an instance within a power domain.

Supply ports and nets can be created in UPF or in the HDL design. If created in the HDL, the port or net shall be of the supply net type defined in the appropriate package UPF (see 11.2). Supply ports shall also be inferred from Liberty using the `pg_pin` attribute (see Annex F). In UPF, supply ports on power-management cells may be specified using the appropriate power/ground options on the define commands (see Clause 7) or specified through Liberty. Supply ports may be specified on macros (see 4.9.2.4) using **create_supply_port**.

4.5.3.2 Supply switches

Supply switches conditionally propagate the value on an input supply port to an output supply port, depending upon the value of a control signal. A supply net is either connected to one or more power switches or supply ports, which function as root supply drivers.

4.5.3.3 Supply sets

A supply set represents a collection of supply nets that provide a complete power source for one or more instances. Each supply set defines six standard functions: **power**, **ground**, **pwell**, **nwell**, **deeppwell**, and **deepnwell**. Each function represents a potential supply net connection to a corresponding portion of a transistor. Each function of a given supply set can be associated with a particular supply net that implements the function.

A global supply set is one that is defined in a given scope and associates supply nets with its functions. One or more local supply sets, called *supply set handles*, can be defined for a power domain, a power switch, an isolation strategy (see 6.45), a level-shifting strategy (see 6.46), or a retention strategy (see 6.50). A supply set can be associated with a supply set handle as a whole; the functions of a supply set handle can be broken out and connected to ports of instances. This association creates a connection between the supply nets represented by corresponding functions of the supply set and supply set handle.

A supply set function is equivalent to a supply net and may be used anywhere a supply net is allowed. The supply set function represents the supply net that is or will be associated with that function of the supply set. The supply set function reference is a symbolic name for the supply net it represents.

A reference to a supply net by its symbolic name is an indirect reference.

NOTE—A supply net may be associated with a function of more than one supply set. The function that a given supply net performs in one supply set is unrelated to the function it may perform in any other supply set.

Published by IEC under licence from IEEE. © 2018 IEEE. All rights reserved.
Documents
Document Sharing Hub
Full document details: BS IEC 61523-4:2023

4.5.4 Supply network construction

4.5.4.1 Introduction

Supply ports and nets are interconnected to create a supply network. Certain definitions and restrictions constrain how these interconnections are made.

4.5.4.2 Supply sources and loads

Supply ports define supply sources and supply loads, as follows:

— The LowConn of an input or inout port is a supply source. The HighConn of an output or inout port is a supply source (including a switch output).

— The LowConn of an output or inout port is a load. The HighConn of an input or inout port is a load (including a switch input).

A port that is neither a top-level port nor a leaf-level port is an internal (hierarchical) port.

4.5.4.3 Supply port/net connections

Connections are made from nets to ports:

a) From a net to (the LowConn of) a port declared in the same scope, or

b) From a net to (the HighConn of) a port declared in a lower scope, or

c) From a net to a pin of a leaf cell

The LowConn of a port may be used as an implicit net and connected to another port.

Only one net connection can be made to the LowConn of a port. Likewise, only one net connection can be made to the HighConn of a port. A source can be connected to a net that is in turn connected to multiple loads.

4.5.4.4 Supply net resolution

A supply net may be unresolved or resolved, as follows:

— An unresolved supply net shall have only one supply source connection.

— A resolved supply net can have multiple supply source connections. The resolution type may restrict how many supply sources can be on at the same time.

A supply net can have any number of load connections.

4.5.4.5 Supply net/supply set connections

Related supply nets can be grouped into a supply set, with each supply net in the group providing one or more functions of the supply set. The supply net corresponding to a given function of a supply set can be specified when the supply set is created or updated (see 6.26). One supply set may be associated with another supply set (see 6.10); this implicitly connects corresponding functions together and therefore it also

Published by IEC under licence from IEEE. © 2018 IEEE. All rights reserved.
Documents
Document Sharing Hub
Full document details: BS IEC 61523-4:2023

implicitly connects the supply nets associated with corresponding functions and any instance ports to which those functions are connected.

4.5.4.6 Supply set function connections

4.5.4.6.1 Overview

Supply functions of a supply set, and the supply nets they represent, can be connected to instances in one of the following ways: explicitly, automatically, or implicitly. Connections are made downward, from ports or nets in the current scope to ports of descendant instances that are in the extent of the domain.

4.5.4.6.2 Explicit and automatic connections

An explicit connection connects a given particular supply set function directly to a specified supply port. See also 6.14 and 6.15.

An automatic connection connects each supply set function to ports of selected instances, based on the *pg_type* of each port, as indicated by the **UPF_pg_type** attribute (see 6.48) or the Liberty pg_type attribute.

For automatic connections, the default connection semantics for each function of a supply set are as follows:

a) **power** is connected by default to ports having the *pg_type* `primary_power`.

b) **ground** is connected by default to ports having the *pg_type* `primary_ground`.

c) **pwell** is connected by default to ports having the *pg_type* `pwell`.

d) **nwell** is connected by default to ports having the *pg_type* `nwell`.

e) **deeppwell** is connected by default to ports having the *pg_type* `deeppwell`.

f) **deepnwell** is connected by default to ports having the *pg_type* `deepnwell`.

4.5.4.6.3 Implicit connections

An implicit connection connects the required functions of a supply set to cell instances that do not have explicit supply ports. Such connections may involve implicit creation of ports and nets, as described in 4.3.9.

Implicit supply set connections are made in each of the following cases:

a) Primary supply set

The functions of a domain's primary supply set are implicitly connected to any instance in the extent of the domain if the instance has no supply ports defined on its interface.

b) Retention supply set

The functions of a retention strategy's supply set are implicitly connected to the state element that implements retention functionality (e.g., a balloon latch, shadow register, or live slave latch) for any register in the domain to which the strategy applies.

c) Isolation supply set

Published by IEC under licence from IEEE. © 2018 IEEE. All rights reserved.
Documents
Document Sharing Hub
Full document details: BS IEC 61523-4:2023

The functions of a supply set for an isolation strategy are implicitly connected to the corresponding isolation cell implied by the application of the strategy.

d) Level-shifter supply sets

The functions of a supply set for a level-shifting strategy are implicitly connected as appropriate to the input, output, or internal supply pins of any level-shifter implied by the application of the strategy.

After UPF-specified power intent has been completely applied, it shall be an error if any instance in the design does not have a supply set function or supply net connected to each of its supply ports, including any implicit power and ground ports.

4.5.4.7 Supply set required functions

Although a supply set represents a collection of six standard supply functions, not all functions are required in every context:

— `power` and `ground` are typically required in all cases.

— `nwell`, `pwell`, `deepnwell`, and `deeppwell` are only required occasionally.

The required functions of a given supply set are determined from its usage and include the following:

a) Any function used to define a power state of the supply set,

b) Any function used for automatic connection of the supply set based on `pg_type`, and

c) Any required function of a supply set handle with which the supply set is associated.

For implementation, a supply net shall be associated with each required function of a supply set. For verification, however, some aspects of the power intent can be verified before associating supply nets with the required functions. A supply set that does not have supply nets associated with each of its required functions is incompletely specified. For any required function of a supply set that is not associated with a supply net, an implicit supply net is created and associated with the function.

4.5.5 Supply equivalence

4.5.5.1 Overview

Various aspects of power management are determined in part by the identity of, and relationships between, supply nets and supply sets. For example, selection of ports to which isolation or level-shifting strategies can be defined based on the identities of the driver and receiver supplies of the sources and sinks connected to a port. Similarly, composition of power domains is possible provided the supplies of the subdomains involved meet certain constraints. In some situations, identical supply nets or supply sets are required; other situations will only require supply nets or supply sets that are equivalent.

There are two kinds of supply equivalence: electrical equivalence and functional equivalence. Functional equivalence can be defined with or without interchangeability.

Electrical equivalence can affect:

— The number of sources of a supply network, and therefore

— Whether resolution is required for that supply network

Electrical equivalence implies functional equivalence with interchangeability, but not vice versa.

Published by IEC under licence from IEEE. © 2018 IEEE. All rights reserved.

Documents
Document Sharing Hub

Full document details: BS IEC 61523-4:2023

Functional equivalence can affect any of the following:

— Insertion of isolation cells, level-shifter cells, and repeater cells

— Legality of power-domain composition

— Validity of driver and receiver supply attributes

— Connection of cells inserted by implementation tools (e.g., repeater cells without any set_repeater directive, tie cells)

— Valid location in definition of isolation and level-shifter cells

The effect that functional equivalence has may be influenced by the value of the -interchangeable in **set_equivalent** command (see 6.44).

Electrical equivalence is primarily related to supply ports and nets. Functional equivalence is primarily related to supply sets.

4.5.5.2 Supply port/net equivalence

Electrical equivalence is determined by connection or declaration, as follows:

a) A port `P` is electrically equivalent to itself.

b) A net `N` is electrically equivalent to itself.

c) If an unresolved net `N` and a port `P` are connected, then `N` and `P` are electrically equivalent.

d) If a resolved net `N` and a port `P` are connected, and `P` is an inout port or is a load of `N`, then `N` and `P` are electrically equivalent.

e) If `A` and `B` are electrically equivalent, and `B` and `C` are electrically equivalent, then `A` and `C` are electrically equivalent.

f) If `A` and `B` are connected via a supply set function (see 4.5.4.5), then `A` and `B` are electrically equivalent.

g) If `A` and `B` are declared electrically equivalent, then `A` and `B` are electrically equivalent.

NOTE—By definition, a port that is a source of a resolved supply net is never equivalent to that resolved supply net because the value provided by the port to the resolved supply net is not necessarily the same as the resolved value of the supply net.

Electrical equivalence implies the two equivalent objects are electrically connected somewhere. If the connection is not evident in the design (e.g., if it is inside a hard macro whose internals are not visible or if it is a connection that is required outside the design), then declaration of electrical equivalence can be used instead of the explicit connection.

An input and the output of a switch are never electrically equivalent; it shall be an error if they are directly connected or declared electrically equivalent. Similarly, the outputs of two different switches are typically not electrically equivalent, unless they are both driving the same resolved net. However, the outputs of two different switches that each drive an unresolved net can still be functionally equivalent if the input supplies of both switches are equivalent, the control inputs of both switches are logically equivalent, and the two switches have the same set of state definitions.

Functional equivalence with interchangeability is determined by connection or declaration, as follows:

— If `A` and `B` are electrically equivalent, then `A` and `B` are functionally equivalent and interchangeable.

Published by IEC under licence from IEEE. © 2018 IEEE. All rights reserved.
Documents
Document Sharing Hub
Full document details: BS IEC 61523-4:2023

— If `A` and `B` are declared functionally equivalent with interchangeability, then `A` and `B` are functionally equivalent and interchangeable.

Functional equivalence without interchangeability is determined by declaration, as follows:

— If A and B are declared functionally equivalent without interchangeability, then A and B are functionally equivalent and non-interchangeable.

Functional equivalence without interchangeability restricts the interchangeability of functionally equivalent supply sets and can typically be used to simplify the definition of power states. The restrictions affect:

— Determination of power-domain lower boundaries (see 6.20).

— Legality of power-domain composition (see 6.16).

— Validity of driver and receiver supply attributes (see 6.48).

— Connection of cells inserted by implementation tools (e.g., repeater cells without any set_repeater directive, tie cells) (see 6.44).

— Insertion of level-shifter cells (see 6.46).

— Valid location in definition of isolation cells and level-shifter cells (see 7.4 and 7.5)

4.5.5.3 Supply set equivalence

A supply set handle is also a supply set.

A supply set function and its associated supply net are electrically equivalent; thus, for purposes of supply net equivalence, a supply set function acts like a supply net.

Corresponding functions of two supply sets are electrically equivalent if:

— their associated supply nets are electrically equivalent, or

— the two supply sets are directly associated with one another, or

— the two supply sets have been decalred electrically equivalent.

Corresponding functions of two supply sets are functionally equivalent and interchangeable if:

— the two supply sets are electrically equivalent, or

— the two supply sets have been declared functionally equivalent and interchangeable.

Corresponding functions of two supply sets are functionally equivalent and non-interchangeable if the two supply sets have been declared functionally equivalent and non-interchangeable.

Two supply sets are electrically equivalent (also functionally equivalent and interchangeable) if:

— they both have the same required functions, and the nets associated with corresponding functions are electrically equivalent; or

— they are associated with each other directly or indirectly via one or more **associate_supply_set** commands (see 6.10); or

— they are each associated directly or indirectly via **associate_supply_set** (see 6.10) with two other supply sets, which are electrically equivalent.

Two supply sets are also electrically equivalent (also functionally equivalent and interchangeable) if they have been declared electrically equivalent; in this case, it shall be an error if they do not have the same required functions.

Published by IEC under licence from IEEE. © 2018 IEEE. All rights reserved.

Documents
Document Sharing Hub

Full document details: BS IEC 61523-4:2023

Two supply sets are functionally equivalent (only) and interchangeable if they both have the same required functions, and the nets associated with corresponding functions are functionally equivalent and interchangeable, with at least one of these nets declared functionally equivalent and interchangeable.

Two supply sets are also functionally equivalent (only) and interchangeable if they have been declared functionally equivalent and interchangeable; in this case, it shall be an error if they do not have the same required functions.

Two supply sets are functionally equivalent and non-interchangeable if they both have the same required functions, and the nets associated with corresponding functions are declared functionally equivalent and non-interchangeable.

Two supply sets are also functionally equivalent and non-interchangeable if they have been declared functionally equivalent and non-interchangeable; in this case, it shall be an error if they do not have the same required functions.

As a consequence of this:

a) Two anonymous supply sets built from equivalent PG functions are equivalent.

b) A supply set and any supply set handle it is associated with are always electrically equivalent.

For filters in power-management strategies (see 4.6.6), two supply sets are PG equivalent if they both have power and ground functions, and the nets associated with corresponding functions are equivalent.

4.5.6 Supply subnets

Supply ports, supply nets, and supply set functions that are electrically equivalent (see 4.5.5) make up a *supply subnet*. A supply subnet that contains no resolved supply nets has a single root supply driver (see 4.5) whose value determines the values of all supply objects in the supply subnet. A supply subnet that contains one or more resolved supply nets can have multiple root supply drivers.

A supply network consists of one or more supply subnets. Two supply subnets are indirectly connected when one contains a supply object that is an input to a resolved supply net and the resolved supply net is in the other subnet, or when one contains a supply object that is an input to a power switch and the output of the power switch is in the other subnet.

The definitions of root supply driver, electrical equivalence, and supply subnet mean that if more than one resolved supply net is present, all equivalent resolved supply nets are part of the same subnet, and all have the same resolution function. This allows all sources of all resolved supply nets to be resolved by one instance of the resolution function. The resolved value is then distributed to any unresolved supply objects in the supply subnet and to any loads of the supply subnet. See 9.2.3 for the simulation semantics of supply networks.

4.5.7 Supply variation

Supply ports, supply nets, and supply set functions take on values that consist of a state and a voltage. Named port states (see 6.4) and named power states (see 6.5) can be defined to represent the nominal voltages (see 3.1) that a supply object may carry. These nominal voltage values are used also for determining whether level-shifting is required (see 6.46).

In an implementation, the actual voltage of an object may vary around the nominal values. There are several sources of such variation. One source of variation is the accuracy of the supply. Supply variation can be modeled in UPF using the **set_variation** command (see 6.54). Supply variation is applied to

Published by IEC under licence from IEEE. © 2018 IEEE. All rights reserved.
Documents
Document Sharing Hub
Full document details: BS IEC 61523-4:2023

nominal voltages to derive variation ranges for those voltages. Other sources of variation are beyond the scope of this standard.

Nominal voltages are not intended to represent library characterization points and should not be used as such. Implementation tools need to take supply variation and other factors into account when determining what library elements to use.

4.5.8 Supply correlation

Supply variation ranges (see 4.5.7) are used when determining whether level-shifting is needed. How such variation ranges are used depends upon whether the supplies are correlated or not.

When the driving and receiving supplies are not correlated, they vary independently. In this case, level-shifting analysis considers the possibility that the two supplies are simultaneously at opposite ends of their respective variation ranges, and therefore compares minimum versus maximum voltage and maximum versus minimum voltage.

When the driving and receiving supplies are correlated, they are assumed to vary consistently, such that if one supply voltage is at its minimum (respectively maximum) value, then the other supply voltage is also at its minimum (respectively maximum) value. In this case, level-shifting analysis only considers the voltage difference when the two supplies are at the same end of their respective ranges, and therefore only compares minimum versus minimum voltage and maximum versus maximum voltage.

Correlation of supplies is transitive: if supplies A and B are correlated, and supplies B and C are correlated, then supplies A and C are also correlated.

4.6 Power management

4.6.1 Introduction

While a power supply network is a static structure, the power delivered via the power supply network can vary over time. Supply sources can provide different voltages; power switches can turn their outputs off or on and can selectively connect different inputs to the output. As a result, the power available to instances in the extent of a power domain will vary, and at any given time, each power domain's supplies will be in one of many possible states. To manage these various states, and in particular to manage the interactions between power domains that are in different states, power management is required.

Power management enables a system to operate correctly in a given functional mode with the minimum power consumption. Adding power management to a design involves analyzing the design to determine which power supplies provide power to each logic element, and if the driver and receiver are in different power domains, inserting power-management cells as required to ensure that neither logical nor electrical problems result if the two power domains are in different power states.

4.6.2 Related supplies

An active component consists of logic elements that receive inputs and drive outputs. The power supplies connected to an active component provide power for this logic. The supply nets that provide power for the logic that receives or drives a given input or output, respectively, are called the *related supplies* of that input or output. Related supplies typically include power and ground supplies and may also include bias supplies.

Published by IEC under licence from IEEE. © 2018 IEEE. All rights reserved.

Documents
Document Sharing Hub

Full document details: BS IEC 61523-4:2023

At the library cell level, related supplies may be identified for each input or output pin of a cell. Each related supply is a supply pin on that cell; the pin typically has a `pg_type` attribute indicating what supply function it provides (primary power, primary ground, etc.). For a cell that has one set of supply connections, all inputs and outputs would have the same set of related supplies. For a cell that has multiple supply connections, such as a cell with a backup power supply, different pins can have different sets of related supplies. This is particularly true of certain power-management cells, such as a level-shifter, which usually has different related supplies for the input and output.

Related supply nets are often considered in a group, as an implicit supply set. An implicit supply set made up of the supply pins of a cell that are the related supplies of a given input or output is by definition equivalent to any supply set that has been connected to those supply pins.

4.6.3 Driver and receiver supplies

Each output of an active component is typically connected to the input of some other active component in the design. The net connecting the two has a driver on one end (the logic driving the output port) and a receiver on the other end (the logic receiving the input). The driving logic is powered by a supply set called the *driver supply*; the receiving logic is powered by a supply set called the *receiver supply*.

The driver supply and the receiver supply can be the same supply set, e.g., if both components are in the same power domain; or the driver supply and the receiver supply can be different supply sets, e.g., if the two components are in different power domains. The driver supply and the receiver supply can also be different, but nonetheless equivalent, e.g., if they are connected externally or if they are generated by supply networks that ensure they always have the same values.

In some cases, the logic driving or receiving a given port is not evident. In particular, the logic inside a hard macro instance may not be represented in a way that can be used by a given tool. Similarly, the logic that drives primary inputs of the design and receives primary outputs of the design is typically not represented as part of the design. In such cases, it is convenient to be able to associate the driver supply or receiver supply of the missing logic with the port that is connected to that logic. UPF defines attributes that can be used to associate this information with ports of a model.

4.6.4 Logic sources and sinks

Logic ports can be a source, a sink, or both, as follows:

— The LowConn of an input or inout logic port whose HighConn is connected to an external driver is a source.

— The HighConn of an output or inout logic port whose LowConn is connected to an internal driver is a source.

— The LowConn of an output or inout logic port whose HighConn is connected to an external receiver is a sink.

— The HighConn of an input or inout logic port whose LowConn is connected to an internal receiver is a sink.

For a logic port that is connected to a driver, the supply of the connected driver is also the driver supply of the port. A primary input port is assumed to have an external driver and therefore is a source; such a port has a default driver supply if it does not have an explicitly defined **UPF_driver_supply** attribute. An internal port that is not connected to a driver is not a source, and therefore, does not have a driver supply in the design. To model this in verification, an anonymous default driver is created for such an undriven port. This driver always drives the otherwise undriven port in a manner that results in a corrupted value on the port.

Documents
Document Sharing Hub
Published by IEC under licence from IEEE. © 2018 IEEE. All rights reserved.
Full document details: BS IEC 61523-4:2023

For a logic port that is connected to one or more receivers, the supplies of the connected receivers are all receiver supplies of the port. A primary output port is assumed to have an external receiver and therefore is a sink; such a port has a default receiver supply if it does not have an explicitly defined **UPF_receiver_supply** attribute. An internal port that is not connected to a receiver is not a sink, and therefore, does not have any receiver supplies.

The following paragraphs define the power management of HDL literals (e.g., 1′b1)

A literal value (e.g., SystemVerilog 1′b1 or VHDL '1') can be connected to an input port of a macro instance either directly or indirectly. The following three cases can be distinguished:

— A literal is directly associated with an input port of a macro.

— A literal is assigned to a wire (in SystemVerilog) or signal (in VHDL) that is then possibly propagated through various port associations and finally associated with an input port of a macro instance, and the initial literal assignment is done within the extent of the power domain in which the macro is instantiated.

— A literal is associated with a formal input port in an instantiation of a module, and that input port is connected to a wire (SystemVerilog) or signal (VHDL) that is then possibly propagated through various additional port associations and finally associated with an input port of a macro instance, and the initial literal port association is done within the extent of the power domain in which the macro is instantiated.

A literal value shall be implemented with a tie cell. A tie cell can be either:

— A primary rail cell, the output of which is supplied by the primary supply of the domain in which the tie cell is located, or

— An Always-On cell, the output which is supplied by a backup supply (i.e., different from the primary supply of the domain in which the tie cell is located).

A literal connected to a macro instance input port with the attribute **UPF_literal_supply** (see 5.6) is modeled as a tie cell instantiated in the domain in which the literal is referenced (which domain also contains the macro instance) and supplied by the specified literal supply.

If the technology involved supports appropriate tie cells and if the specified literal supply is available in terms of supply availability rules (see 6.20), the implementation shall be a tie cell in the domain. It shall be an error if the technology involved does not support appropriate tie cells (e.g., no support of Always-On tie cells whereas the specified literal supply is different from the primary supply of the domain in which the tie cell is located) or if the specified literal supply is not available. If the technology involved does not support any tie cells, the implementation shall be a connection to the appropriate rail of the specified literal supply.

A literal connected to a macro instance input port that does not have the attribute **UPF_literal_supply** is modeled as a tie cell instantiated in the domain in which the literal is referenced (which domain also contains the macro instance) and supplied by the receiving supply of the macro port, defined by the predefined attribute **UPF_receiver_supply,** or by predefined attributes **UPF_related_power_port**, **UPF_related_ground_port**, and **UPF_related_bias_ports** (see 5.6).

If the technology involved supports appropriate tie cells and if the receiving supply is available in terms of supply availability rules (see 6.20), the implementation shall be a tie cell in the domain. It shall be an error if the technology involved does not support appropriate tie cells (e.g., no support of Always-On tie cells whereas the receiving supply is different from the primary supply of the domain in which the tie cell is located) or if the receiving supply is not available. If the technology involved does not support any tie cells, the implementation shall be a connection to the appropriate rail of the receiving supply.

For any other cases, a literal is modeled as a tie cell instantiated in the domain in which the literal is referenced and supplied by the primary supply of this domain.

Documents
Document Sharing Hub
Published by IEC under licence from IEEE. © 2018 IEEE. All rights reserved.
Full document details: BS IEC 61523-4:2023

If the technology involved supports tie cells, the implementation shall be a primary rail tie cell in that domain. If the technology involved does not support any tie cells, the implementation shall be a connection to the appropriate rail of the primary supply.

4.6.5 Power-management requirements

Power management is required to mediate the changing power states of power domains in the system and the interactions between power domains that are in different states at various times. There are four specific areas addressed by power management, as follows:

— If a power domain is powered down in certain situations, its state registers may need to have their values saved before power-down and restored after subsequent power-up, either to maintain persistent data or to enable faster power-up.

— If the distance between driver and receiver is long (the capacitive load is high), buffers (repeaters) may be required to strengthen the signal along the way, or to ensure that it stabilizes within the required time.

— If a receiver is powered on, but its driver is not, an isolation cell is required between driver and receiver to drive the receiver with a known value despite the fact that the ultimate driver is powered off.

— If the driver and receiver supplies (or isolation and receiver supplies, or driver and isolation supplies, etc.) are operating at different voltage levels, a level-shifter is required between them to translate between voltage levels.

UPF provides commands for specifying where power-management structures should be added to a design to address each of these areas.

4.6.6 Power-management strategies

Addition of power-management cells to a design is driven by rules or strategies. UPF provides commands for specifying retention strategies (see 6.50), repeater strategies (see 6.49), isolation strategies (see 6.45), and level-shifting strategies (see 6.46). Each of these strategies can be defined in various ways to apply to specific design features or more generally to classes of features. Precedence rules (see 5.7) define how multiple strategies for the same feature are to be interpreted. In general, more specific strategies take precedence over more general strategies.

Retention strategies apply to specific state variables in a given power domain or to all state variables in a domain. A retention strategy also defines the power supplies, the control signals and their interpretation, and certain behavioral characteristics of the retention registers to be used for the state variables to which it applies.

Repeater, isolation, and level-shifting strategies apply to ports of a power domain. The ports to which one of these strategies applies can be defined by name or can be selected by filters. Source and sink filters select ports based on the driver supply and receiver supply, respectively, of each port. The filters typically match exact named supplies, electrically PG equivalent supplies and, if specified, functionally PG equivalent supplies. Ports can also be selected by direction. Each of these strategies also specifies the relevant power supplies and control signals and their interpretation to be used for any power-management cells added by the strategy.

4.6.7 Power-management implementation

Implementation of power-management strategies involves adding power-management cells—retention registers, repeaters (buffers), isolation cells, and level-shifter cells—to the design. Each added cell may add

Documents
Document Sharing Hub
Published by IEC under licence from IEEE. © 2018 IEEE. All rights reserved.
Full document details: BS IEC 61523-4:2023

new driving and receiving logic and as a result may change the driver and receiver supplies of a given port, which could potentially affect the application of other strategies based on source and sink filters. To ensure the interaction of multiple strategies is well defined, strategies are applied according to the following rules:

a) Strategies are implemented in the following order: retention strategies, followed by repeater strategies, followed by isolation strategies, followed by level-shifter strategies.

b) A retention strategy may affect the driving supply of the retention cell output. If so, the new driving supply of the retention cell is visible to, and affects the result of, a source filter of any subsequently applied strategy.

c) A repeater strategy causes insertion of a buffer, which has a receiver and a driver; this insertion therefore affects both the receiving supply of ports driving the repeater input and the driving supply of ports receiving the repeater output. The new driving supply and receiver supply are visible to, and affect the result of, source and sink filters, respectively, of any subsequently applied strategy.

d) An isolation strategy may cause insertion of an isolation cell, which has a receiver and a driver; therefore if such insertion occurs, it affects both the receiving supply of ports driving the isolation cell input and the driving supply of ports receiving the isolation cell output. However, the new driving supply and receiver supply are not visible to, and do not affect the result of, source and sink filters, respectively, of any subsequently applied isolation or level-shifting strategies.

e) A level-shifting strategy may cause insertion of a level-shifting cell, which has a receiver and a driver; therefore if such insertion occurs, it affects both the receiving supply of ports driving the level-shifting cell input and the driving supply of ports receiving the level-shifting cell output. However, the new driving supply and receiver supply are not visible to, and do not affect the result of, source and sink filters, respectively, of any subsequently applied level-shifting strategy.

Repeater, isolation, and level-shifting strategies apply to all ports on the interface of a power domain, both those on the upper boundary of the domain and those on the lower boundary of a domain. As a result, a port on the boundary between two domains—the upper boundary of one, and the lower boundary of the other—may have multiple strategies of a given type defined for it, one from each of the two domains. In such a case, both strategies may cause addition of power-management cells.

4.6.8 Power-management cells

Power-management cells that have a single set of supply connections are also referred to as *single-rail power-management cells*. Single-rail cells typically share the same supply as the domain primary, but can also be physically placed in a different supply region. A typical example of a single-rail cell is an isolation cell placed in the destination domain.

Power-management cells that have two sets of supply connections are also referred to as *dual-rail power-management cells*. Dual-rail cells have a secondary or backup supply that enable them to be placed in the primary domain but still have secondary supply connectivity to other supplies. Typical examples of dual-rail power-management cells are dual-rail buffers and inverters, isolation cells placed in switched source domain, retention flops, power switches, level-shifters, etc.

Power-management cells with more than two sets of supply connections are also referred to as *multi-rail power-management cells*. Multiple supplies are more common in macros than standard cells. Typical examples of this case are multi-rail level-shifters (input supply, domain supply, output supply), power multiplexors, etc.

Documents
Document Sharing Hub
Published by IEC under licence from IEEE. © 2018 IEEE. All rights reserved.
Full document details: BS IEC 61523-4:2023

4.6.9 Power control logic

Most power-management cells require control signals to coordinate their activity. In particular, isolation cells require enable signals, retention cells may require save and restore signals or related control inputs, and power switches (see 4.5.3.2) require switch control signals. Logic ports and nets that implement these control signals may be present already in the HDL design or they may be added via UPF commands.

Control logic ports and nets defined in UPF are created within the logic hierarchy independent of power-domain definitions. This allows the power control network to be created and distributed across power domains.

A control signal is logically equivalent to itself. Two different control signals are logically equivalent to each other if one is directly connected to the other, if they are both directly connected to a common logic source (see 4.6.4), or if their respective logic sources are equivalent.

4.7 Supply states and power states

4.7.1 Overview

Power is required for the operation of a system. Supply ports, supply nets, and supply set functions propagate power from root supply sources to the active components of a system. Supply switches affect the propagation of power from supply sources to supply consumers. Active components consume power as required by a given operating mode.

An object that propagates power can be in various *supply states*. A supply state of an object that propagates power represents the power provided by the supply source(s) of that object and therefore the power it can propagate to power-consuming objects to which it is connected, assuming an ideal power source that can handle an infinite load.

An object that consumes power can be in various *power states*. A power state of an object that consumes power represents an operating mode of that object and therefore the power required by the object in that operating mode.

Power states can also be defined for collections of objects. Such power states name combinations of power states and/or supply states of other objects and may impose constraints on such combinations.

4.7.2 Supply states

The supply states of supply ports, supply nets, and supply set functions are represented by type **supply_net_type**, defined in package UPF (see 11.2). This type models electrical values as a combination of two values: a state value and a voltage value, which together constitute the supply state of a supply port, supply net, or supply set function.

— The state value is one of OFF, UNDETERMINED, PARTIAL_ON, or FULL_ON. The state value represents the ability of the object's root supply source(s) to provide power.

— The voltage value is internally represented as an integer number of microvolts, measured relative to a single common reference ground that is assumed to apply to the entire design. The voltage value is relevant only for the PARTIAL_ON and FULL_ON state values; it is undefined for the OFF and UNDETERMINED state values.

— The state value is not affected by or determined by the voltage value.

Published by IEC under licence from IEEE. © 2018 IEEE. All rights reserved.
Documents
Document Sharing Hub
Full document details: BS IEC 61523-4:2023

Supply states of a supply port that is a supply source propagate to any supply net or supply set function to which that port is electrically equivalent.

Supply states of a supply port are defined as named *port states*. Port states may be referenced in a power state table to specify legal combinations of port states that may exist.

Supply switches also have named states, which correspond to control expressions that determine which input(s) of a switch affect the switch output supply state (see 4.5.3.2). The named states of a supply switch therefore determine whether and how the switch propagates the supply state(s) of its input supply port(s) to its output supply port.

4.7.3 Power states

Power states are defined for supply sets, power domains, composite domains, groups, modules, and instances.

By default, any object for which power states can be defined has a predefined power state UNDEFINED. This power state initially represents the undifferentiated set of all possible functional states of that object. Specific states within that set of all possible power states of the object can be defined. Defining a power state creates a *named power state* that represents a subset of the functional states of the object. Defining a named power state removes that subset of functional states from the set of functional states represented by the UNDEFINED power state.

A given power state is *active* when certain conditions occur. When a power state of an object is active, that power state characterizes the operating mode of the object at that time. More than one power state of a given object may be active at the same time, subject to certain restrictions.

A power state of an object is characterized by its *defining expression*. The defining expression for a named power state is the logic expression specified in its definition (see 6.5). The defining expression of the UNDEFINED power state is effectively the condition that no other power state of this object is active.

A named power state is a *definite power state* if its defining expression consists of a single term or conjunction of terms, such that each term is one of the following:

a) a Boolean expression using bit-wise conjunction or logical conjunction over signals in the design, or

b) a term of the form "<object>==<state>", where

   1) <object> is the name of an object for which power states are defined and

   2) <state> is the name of a definite power state of <object>.

In the latter case, the term evaluates to True when <state> of <object> is active. A definite power state is active when its defining expression evaluates to True.

A named power state is a *deferred power state* if it has no defining expression. This can occur when the exact definition will involve implementation details that are not yet known, such as which supply rail will be switched, or what control signals will determine the state. A deferred power state is considered to be a definite power state whose defining expression will be provided at a later time. A deferred power state is active when the state has been assigned to an object by the `set_power_state` function, or when certain conditions occur for predefined power states (see 4.7.4).

A named power state that is neither a definite power state nor a deferred power state is an *indefinite power state*. An indefinite power state is active when its defining expression evaluates to True.

Published by IEC under licence from IEEE. © 2018 IEEE. All rights reserved.

Documents
Document Sharing Hub

Full document details: BS IEC 61523-4:2023

A named power state R of an object O is a *refinement* of another named power state S of the same object O if both R and S are definite states and the defining expression of R includes exactly one term of the form O==S. A named power state S of an object O is an *abstraction* of another named power state T of the same object O if T is a refinement of S. The refinement and abstraction relations are both transitive; if R is a refinement of S, and S is a refinement of T, then R is a refinement of T, and T is an abstraction of R.

If one power state is an abstraction or refinement of another power state, then the two power states are *related by refinement*. For a given set of power states that are related by refinement, the *most-refined power state* is the unique state that is not an abstraction of any other state in the set. A power state S of an object is a *fundamental power state* if it is a power state that is not a refinement of any other power state of that object.

By definition, if power state R is a refinement of power state S, and R is active, then S is also active. Also by definition, no named power state is related by refinement to the **UNDEFINED** power state.

Fundamental power states of a given object shall be *mutually exclusive*. It shall be an error if two fundamental power states of the same object are both active at the same time. Similarly, two different refinements of the same power state shall be mutually exclusive. It shall be an error if two different refinements of the same power state are both active at the same time. The predefined power state ERROR represents the error condition in which two states that should be mutually exclusive are both active at the same time.

A power state can be either *legal* or *illegal*. A legal power state represents a state of an object that is intended or expected to occur in normal operation of the system. An illegal power state represents a state of an object that is not intended or expected to occur in normal operation. By default, a named power state is legal unless its definition specifies that it is illegal or it is a refinement of an illegal power state. Therefore a legal state may be an abstraction of an illegal state, but an illegal state cannot be an abstraction of a legal state. Equivalently, a legal state may have an illegal refinement, but an illegal state cannot have a legal refinement.

A power state that is legal for a given model may be marked as illegal for a given instance of that model. Any refinement of such a power state is also illegal for that instance of the model.

The *current power state* of an object is determined as follows:

— If exactly one named power state of the object is active

— Then that state is the current power state; else

— If all active states of the object are definite states that are related by refinement

— Then the most-refined power state is the current power state; else

— The predefined ERROR state is the current power state

The set of power states for a given object may be marked as *complete*, which indicates that all fundamental states of the object have been defined as named power states. If the set of power states for an object is complete, then it shall be an error for the **UNDEFINED** power state to be the current power state of that object. It is also an error if a new fundamental power state is defined after the power states are marked complete.

NOTE 1—By definition, a fundamental power state of an object is active whenever any refinement of that power state is active.

NOTE 2—Three distinct error conditions can occur related to active or current power states:

1) An illegal power state of an object is active.

Published by IEC under licence from IEEE. © 2018 IEEE. All rights reserved.
Documents
Document Sharing Hub
Full document details: BS IEC 61523-4:2023